Generating Referring Expressions in a 3D Environment

Final Project for the Degree in Computer Science

Adrián Rabadán Jurado
Teresa Rodríguez Ferreira

Directed by

Raquel Hervás Ballesteros
Gonzalo Méndez Pozo

Departamento de Ingeniería de Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Madrid, 23 de Junio de 2014

Generating Referring Expressions in a 3D Environment

Final Project

Adrián Rabadán Jurado
Teresa Rodríguez Ferreira

Directed by

Raquel Hervás Ballesteros
Gonzalo Méndez Pozo

Departamento de Ingeniería de Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Madrid, 23 de Junio de 2014

To my friend and work partner for these past three years,
and hopefully for many more.

Agradecimientos

En primer lugar, queremos dar las gracias a Raquel y a Gonzalo por guiarnos y ayudarnos como lo han hecho. Por trabajar con nosotros, y por leerse y releerse esta memoria tan larga.

También a todos aquellos familiares, amigos y conocidos que han respondido a nuestras encuestas y han hecho posible el estudio.

Agradecemos también que otras personas tuvieran nuestros problemas antes que nosotros, y que los resolviesen en los foros.

Autorización

Se autoriza a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria como el código, la documentación y/o el prototipo desarrollado.

Adrián Rabadán Jurado

Teresa Rodríguez Ferreira

Resumen

Desde hace muchos años, el ser humano ha querido entender a las demás personas y su forma de pensar. Esto no resulta sencillo, ya que cada individuo, debido a condiciones como el sexo, la edad, la lengua materna, la cultura o los conocimientos y las vivencias previas, analiza las cosas de forma distinta. Algunas de las consecuencias de las diferentes formas de ver el mundo se ven reflejadas a la hora de la comunicación.

En este proyecto se ha investigado cómo las personas componen descripciones para referirse a alguien en concreto, y se han desarrollado algoritmos que son capaces de reproducir este comportamiento. En el campo de la Generación de Expresiones de Referencia ya hay muchos avances documentados y experimentos diversos, y este proyecto se ha centrado en investigar el comportamiento de las personas al analizar situaciones reales y cotidianas. Se ha creado una aplicación que reúne este conocimiento, y se ha demostrado que funciona proponiendo situaciones cargadas de objetos y personajes. Con los algoritmos creados, se han realizado descripciones fáciles de entender (usando lenguaje natural como lo haría un humano) que permiten reconocer personajes de forma rápida, y que tienen un bajo porcentaje de fallo para cualquier tipo de persona. Para ello se han considerado las características de la situación y del personaje a describir, y se han diseñado algoritmos que se adaptan a ellos.

Para generar las mejores situaciones para poner a prueba a nuestros algoritmos, se ha creado con el motor Unity 3D una habitación cerrada y llena de personas, como es la cafetería de la Facultad de Informática, y se ha empleado una gran variedad de personajes suficientemente diferenciables en cada situación. Con ello se han podido cronometrar las respuestas de usuarios reales y se ha determinado qué descripciones funcionan mejor en determinadas situaciones. Con la información recopilada tras pruebas reales con voluntarios, se han afinado aún más los algoritmos, creando el tipo de descripciones que personas reales quieren oír.

Abstract

For years, humans have tried to understand themselves and their way of thinking. This is not easy, because each person, due to factors such as their sex, age, native language, culture or their knowledge and previous experiences, analyses things differently. Some of the consequences of these different points of view are reflected in how we communicate.

In this project we have researched the way in which people produce descriptions to refer to someone in particular, and the details they notice in specific situations. We have developed algorithms that are able to reproduce this behaviour. There is already a lot of progress in the field of the Generation of Referring Expressions documented in books and experiments, and this project focuses on analysing people's descriptions in real life, everyday situations. We have created an application that uses all this knowledge, and we have demonstrated that it works by creating situations filled with objects and characters. With the algorithms we have designed, we have built descriptions which are easy to understand (using natural language in the way that a human would) and which allow the user to recognise any character easily with a low failure rate for any type of person. For this purpose, we have considered the details of the situation and of the character that has to be described, and we have designed algorithms that adapt to them.

In order to generate the best situations to test our algorithms, we have created a large closed room full of people with the 3D engine Unity. This room is the canteen in the IT building in our university, and we have used a large variety of characters that are different enough from one another. With this we have been able to time the answers of real users and determine which descriptions work best in different kinds of situations. With the information we gathered from our surveys with volunteers, we have been able to improve our algorithms even more, creating the kind of descriptions that real people like to hear.

Palabras Clave

- Algoritmos
- Descripciones
- Entorno 3D
- Generación de Expresiones de Referencia
- Generación de Lenguaje Natural
- Unity

Keywords

- 3D environment
- Algorithms
- Descriptions
- Generation of Natural Language
- Referring Expression Generation
- Unity

Contents

Index of figures

Capítulo 1

Introducción

En todas las conversaciones los humanos nos referimos a personas, objetos, lugares y situaciones. Cada vez que lo hacemos inconscientemente mencionamos distintas propiedades suyas, que permiten a los demás entender a qué nos referimos. Pero, ¿cómo elegimos esas propiedades? Normalmente las personas no se dan cuenta de que hay ciertos patrones para sus descripciones. Propiedades que resaltan más para cada situación o incluso para cada persona en particular.

Algunas personas dicen que se fijan mucho en si un hombre lleva o no barba, mientras que otros no se dan cuenta. Otros se centran en el tipo de ropa que lleva la persona, mientras que otros le dan más importancia al color de su ropa, y no al tipo. Cada uno tiene su propia forma de ver el mundo, y su propia forma de describirlo. Nosotros, sin embargo, estamos interesados en encontrar las características comunes en las que todas las personas se fijan, los patrones que se pueden aplicar a la forma que tiene la mayoría de la gente de producir una descrpipción.

1.1. Motivación

Es muy importante poder generar descripciones para personas o para objetos. Al dar indicaciones a una persona, al contar una historia o simplemente al hablar sobre otra persona, tenemos que ser capaces de describirles para que sepan a quién o a qué nos referimos. Poder generar estas descripciones de forma automática, con una aplicación, puede ser muy útil. Por ejemplo podría usarse para dar indicaciones sobre cómo llegar a un sitio. En vez de instrucciones como *"Camina hacia delante cien metros y después gira a la derecha"*, que no son muy útiles cuando se va andando, podría decir *"Camina hasta el hombre de la camisa amarilla y después gira a la derecha"*. En el futuro este tipo de aplicación podría integrarse con Google Glass para escanear y analizar los alrededores de la persona,

y ofrecerles una descripción. Este último ejemplo podría suponer un gran avance para personas con problemas de visión.

Cuando la generación artificial de estas descripciones se lleva a aplicaciones reales, es importante que las descripciones suenen naturales y lo más cercanas posible a la forma de hablar de un humano. Por este motivo ha sido importante para nosotros no solo crear descripciones precisas con nuestros algoritmos, sino también hacer que estas descripciones sean naturales y realistas.

El objetivo de este proyecto es la generación de buenas descripciones, o expresiones de referencia, que permitan identificar correctamente a una persona en una escena. Estas expresiones de referencia permiten al usuario distinguir a esa persona del resto de gente y objetos de la escena. Imitan el lenguaje natural, y suenan lo más parecido posible a lo que diría un humano. La aplicación genera todas las descripciones en inglés, el idioma más comunmente utilizado del mundo, y después son traducidas manualmente a español, para que los usuarios puedan utilizar esta aplicación en cualquiera de los dos idiomas.

Ha habido mucho progreso en este campo, pero lo que diferencia a nuestro proyecto es el hecho de que hemos probado todo nuestro trabajo con usuarios reales, y hemos creado situaciones que están totalmente basadas en lugares y personas existentes. Con este proyecto esperamos contribuir al campo de la Generación de Expresiones de Referencia.

1.2. Objectivos

A continuación enumeramos los objetivos que queremos cumplir:

- Identificaremos los detalles que son importantes para la mayoría de la gente en las descripciones y el orden en el que los suelen mencionar
- Crearemos distintos algoritmos que generen expresiones de referencia precisas
- Combinaremos estos algoritmos para crear un meta-algoritmo que ofrezca la descripción más apropiada para cada situación
- Crearemos un entorno 3D para probar los algoritmos

1.3. Método de trabajo

Hay tres encuestas distintas en ese proyecto. Cada una de ellas fue enviada a los usuarios después de terminar una porción del trabajo. Las encuestas nos ayudan a decidir cómo abordar la siguiente parte del trabajo a realizar. Ya que la información que se incluye en una descripción es un tema muy subjetivo, primero tenemos que descubrir cómo los usuarios componen sus descripciones. Las encuestas también nos ayudan a comprobar que el trabajo que ya está terminado es correcto. Con los resultados que conseguimos, podemos encontrar fallos que tenemos que corregir o partes del trabajo que se pueden mejorar.

La primera encuesta fue enviada después de haber tomado solo algunas de las decisiones de diseño. No teníamos una idea clara de las propiedades que tenían que aparecer en las descripciones, y nuestros usuarios nos ayudaron a descubrirlas. La segunda encuesta nos permitió comprobar que nuestros algoritmos eran eficaces, y también nos ayudó a decidir cómo implementar el meta-algoritmo. Por último, la última encuesta sirvió para ver si nuestra aplicación realmente funcionaba correctamente, y nos dio ideas de posibles modificaciones futuras.

1.4. Estructura del documento

Empezamos estudiando la información e investigación existente en el campo de la Generación de Expresiones de Referencia. El capítulo 3 está dedicado a revisar y analizar distintos algoritmos, algunos bastante sencillos y otros más complejos.

En el capítulo 4 empezamos haciendo los preparativos necesarios para el proyecto. Tomamos algunas decisiones importantes de diseño relativas al lenguaje que vamos a usar para la programación, el comportamiento de nuestra aplicación y la forma en la que el usuario podrá interactuar con ella.

En el capítulo 5 analizamos los resultados de nuestra primera prueba con usuarios. Esta encuesta emplea fotografías tomadas en la cafetería de nuestra facultad y pide a los usuarios tanto que identifiquen a ciertas personas dada su descripción, como que escriban sus propias descripciones.

En el capítulo 6 explicamos en detalle los algoritmos que hemos creado y su comportamiento. También describiremos las escenas que hemos creado y cómo hemos modificado a los personajes para conseguir variedad. Explicamos cómo hemos almacenado la informacón de cada personaje y cómo la aplicación usa esta información para generar las expresiones de referencia.

En el capítulo 7 analizamos los resultados de nuestra segunda prueba con usuarios. Para estas pruebas, les ofrecemos fotografías de las escenas y personajes creados en Unity. Primero les pedimos que identifiquen a los personajes a los que nos referimos, y después les pedimos que valoren las descripciones generadas por nuestros algoritmos para que podamos comprobar su eficacia.

En el capítulo 8 hacemos las últimas modificaciones a nuestra aplicación basándonos en los resultados obtenidos en la encuesta anterior. Además creamos un meta-algoritmo que combina los algoritmos anteriores, y produce diferentes expresiones de referencia dependiendo de la naturaleza de la escena y del personaje descrito.

La última encuesta es analizada en el capítulo 9. Pedimos a nuestros usuarios que prueben la aplicación y almacenamos información sobre sus respuestas y el tiempo que tardaron en encontrar a cada personaje. Esto nos ayuda a evaluar lo eficaces que son realmente nuestros algoritmos.

El trabajo realizado por cada miembro del grupo está detallado en el capítulo 10. Describimos el trabajo que hemos hecho entre los dos y el trabajo realizado de forma individual.

Finalmente, revisamos y discutimos los resultados obtenidos con este proyecto en los capítulos 11 y 12.

Hemos incluido dos anexos que detallan las instrucciones que se deben seguir para modificar la aplicación, y también el manual de usuario y la guía de instalación.

Chapter 2

Introduction

In every conversation, human beings refer to people, objects, places and situations. Every time we do this, we unconsciously mention different aspects or properties that they have and allow others to understand what we are referring to. But how do we choose these properties? Usually people do not realise that there are certain patterns to their descriptions, certain properties that stand out more for each situations or even for each person in particular.

Some people will say that they pay special attention to whether a man has a beard or not, while some do not notice. Some focus on the type of clothes that someone is wearing, while others notice the colour they are dressed in, but not the type of their clothes. Every person has their own way of looking at the world, and their own way of describing it. However, we are interested in finding the common aspects that everybody notices, the patterns than can apply to how most people give a description.

2.1. Motivation

It is very important to be able to generate descriptions for people and objects. When giving directions to a person, when telling a story, or simply talking about someone else, we need to be able to describe them so that the hearer knows who or what we are referring to. Being able to generate these descriptions artificially, with an application, can be very useful. For example, it can be used to provide instructions to get to a certain place. Instead of offering directions like *"Walk forward for one hundred meters and then turn right"*, which are not very useful when walking, it could say *"Walk towards the man wearing a yellow shirt and then turn right"*. In the future this type of application could be integrated with Google Glass to scan and analyse the person's surroundings and then provide descriptions to them. This example would be a huge step forward for people who are visually

impaired.

When the artificial generation of these descriptions is taken to real applications, it is important for the descriptions to sound natural and as close as possible as a human's way of speaking. For this reason it has been important for us not only to create accurate descriptions with our algorithms, but also to make these descriptions seem natural and realistic.

The purpose of this project is to generate good descriptions, or referring expressions, that correctly identify a certain person inside a large scene. These referring expressions allow the user to distinguish that person from the rest of the people and objects in the scene. They imitate natural language, and sound as close as possible to what a real human would say. The application generates all the descriptions in English, the most commonly used language in the world, and they are then translated into Spanish manually, so users can use this program in either language.

There has been a lot of progress in this field, but what makes our project different is the fact that we have tested all our work on real users and we have created situations that are based completely on existing locations and people. With this project we hope to contribute to the field of the Generation of Referring Expressions.

2.2. Objectives

The following are the objectives that we wish to achieve:

- We will identify the details that are important for most people in their descriptions, and the order in which they mention them
- We will create different algorithms that generate accurate referring expressions
- We will merge these algorithms together to create a meta-algorithm that offers the most appropriate description for the situation
- We will build a 3D environment to test our algorithms in

2.3. Work Method

There are three different surveys in this project. Each of them was sent to the users after a portion of our work had been completed. Every survey helps us decide how to approach the next section of work that needs to be done. Since the information contained in a description is a very subjective

matter, we first need to find out how our users compose descriptions. The surveys also help us check that the work which we have already finished is well received by the users. From the results we gather, we can find mistakes that we have made or parts of our work which need to be improved.

The first survey was sent out after only a few design choices had been made. We did not have a clear idea of the properties that had to be included in the descriptions, and our users helped us discover them. The second survey helped us check whether our algorithms were effective, and it also allowed us to decide how to implement the meta-algorithm. Finally, the last survey served to see how well our application was really working, and it also gave us clues for possible future improvements.

2.4. Document Structure

We start by studying the existing information and research in the field of the Generation of Referring Expressions. Chapter 3 is dedicated to reviewing and analysing different algorithms, some quite basic and others more complex.

In chapter 4 we start making the necessary preparations for the project. We make some important design decisions concerning the language used for the code, the behaviour of our application and the way the user will be able to interact with it.

In chapter 5 we analyse the results of our first test with users. This test makes use of photographs taken in the canteen in our university and asks users both to identify certain people given their descriptions, and also to write their own descriptions.

In chapter 6 we explain in full detail the algorithms we have created and their behaviour. We also describe the scenes we have generated and how we have modified the characters to achieve variety. We explain how we have stored each characters' information and how the application uses this information to generate the referring expressions.

In chapter 7 we analyse the results of our second test with users. For these tests we provide them with photographs of the scenes and characters created in Unity. First we ask them to identify the characters we are referring to again, and then we ask them to rate the descriptions generated by our algorithms so we can compare their effectiveness.

In chapter 8 we make the last modifications to our application based

on the results obtained from the previous survey. We also create a meta-algorithm that combines the previous algorithms, and produces different referring descriptions depending on the nature of the scene and the character being described.

The last survey is analysed in chapter 9. We ask our users to test the final application and store information about their answers and the time it takes them to identify each character. This helps us assess how effective our algorithms really are.

The work that has been done by each of the members of the group is detailed in chapter 10. We describe the work that we have done together and the work that has been done individually.

Finally, we review and discuss the results we have obtained in this project in chapters 12 and 11.

We have included two annexes that detail the procedure that should be followed in order to modify this application, and also the user manual and installation instructions.

Chapter 3

State of the Art

In this section we will briefly review the fields of Natural Language Generation and Referring Expression Generation. We will explain what it consists in and why it is important. We will also discuss some of the most important algorithms created over the years and compare them. Finally we will explain the conclusions that we have reached and how this existing work will influence our own research.

3.1. Natural Language Generation

Natural Language Generation (NLG) is one of the tasks of Natural Language Processing, which is one of the many fields of Artificial Intelligence. It consists in producing natural language from a machine representation of data, such as tables of numerical data or entries from a database. A program that performs this task might need to generate understandable text from data, or it might need to re-phrase text in order to make it more readable. This field has been studied since the 1970s, and has many applications. NLG applications can generate textual summaries from data-sets, so an example of application would be a weather forecast generator. It can receive a series of data as input, and generate a description of the following day's weather as output, and research (Turner et al., 2009) has even shown that some people prefer these computer-generated descriptions to human-generated text.

NLG systems have a series of stages (Reiter and Dale, 2000; Hervás, 2009) that are usually followed:

1. Content Determination: The first step, selecting the information that should be included in the text.

2. Discourse Planning: Organising the information in each sentence and planning the structure of the text.

3. Aggregation: Merging related sentences and determining how compact the information should be.

4. Referring Expression Generation: Producing descriptions for objects that the text refers to.

5. Lexicalisation: Selecting the words that will be used to represent concepts in the text.

6. Surface Realisation: Creating the final text, obeying the rules of syntax, morphology and spelling.

The field of NLG is very wide, but in our research we will focus on Referring Expression Generation. Even though this is just one of its subtasks, there is a huge amount of research on the topic, and it is a very important step in the Generation of Natural Language. Every time an object, person, place or idea is mentioned, it must be described in a way that can be understood by a person. These descriptions can be as simple as a single noun, or they can be complex sentences. Here we explore the research that has already been done in this field, so we can use it as a starting point in our own work.

3.2. Referring Expression Generation

The field of referring expression generation (REG) has been studied for over forty years. A first approach, a primitive incremental algorithm, was described in 1972 (Winograd, 1972). Since then there have been many different approaches to the problem of generating a description for an object or person. However, they all have something in common: the main difficulty of generating a good description is choosing which properties of the item being described should be mentioned and in what order. Here we will explain some of the most significant algorithms created in this field. They are not all implemented in our project, but they have all been very important in the history of Natural Language Generation. Some of the information about the algorithms gathered in the following sections is based on the work of Krahmer and Van Deemter (2012).

3.2.1. What is a good referring expression?

A referring expression is a description created with the intention of distinguishing a certain object or person (referent) from a number of other objects or people (distractors). It must identify the referent unambiguously,

effectively ruling out all the distractors.

In order to be a good referring expression, it must also obey Grice's maxim of Quantity (Grice, 1975) (*"Do not make your contribution more informative than is required"*). This means that the description should not include any unnecessary information. For example, if there is only one man in it, a good description would not be *"The man in the red shirt"*. This might correctly identify the intended referent and rule out everybody else in the room, but a person who hears this description might infer that the fact that he is wearing a red shirt is important. Otherwise it would not have been mentioned since it provides no useful information.

A good referring expression should also not include information that is too specific. For instance, if we have a room with only a woman in it, and she is described as *"The teacher"*, the hearer would probably expect her to be described as *"The woman"*. The fact that she is described as a teacher might make the hearer imply that it is somehow important that this woman is a teacher.

Another factor that has been considered important in the past is local brevity (Dale, 1989a). This means that the description generated should be the shortest one possible. Most algorithms, however, do not generate a minimal description. As Dale and Reiter discussed, people usually use descriptions that include more information than is strictly necessary, so local brevity is not considered in most algorithms.

3.3. REG Algorithms

Many different algorithms have already been designed, and there is not one "right" way to do it. They all have interesting approaches and they all contribute to the field. Some of these algorithms, however, have been used for years as the basis for further work, because they are simple and effective. All these basic algorithms were created before the year 2000, and since then, more complex algorithms have been used, but for our purpose we will use these basic ones as our starting point.

Our approach is based on attribute-value pairs. These algorithms consider that items have properties or *attributes*, and these attributes have *values* that distinguish that particular item. So for instance, a pencil might have the attributes *"colour"*, *"length"* and *"thickness"*, with the values *"black"*, *"10"* and *"medium"* respectively. We will also study Relational Algorithms. These algorithms don't focus on the properties of a particular object, instead they study the relation that the object has with the rest of the objects in the scene.

With these algorithms that we have chosen to base our project upon, we cannot cover all the work that has already been done in this field, but we will be able to create a sturdy program, that mixes these different approaches in a way that has not been done before.

When faced with the problem of generating a good referring expression, we start off with a context set (the entities in the current scene), the contrast set (all the entities except the one that has to be described), the referent (the entity we are describing), the distractors (each of the other entities that still fit into the description generated so far), a set of attributes for each entity (the aspects that can be described, such as hair colour or posture) and a value for each attribute (for instance *"blue"* or *"standing up"*). The resulting referring expression will be a collection of attribute-value pairs in a certain order, which can later be rendered into natural language.

Every entity has a special attribute known as "type". This will be the noun used to refer to it, and it will always appear in the referring expression, even if it does not rule out any of the distractors, because people tend to use the referent's type in virtually all their descriptions.

Different algorithms differ in the way they choose which attributes should be included in the referring expression.

3.3.1. Full Brevity Algorithm

The Full Brevity Algorithm (Dale, 1989a) always generates the shortest possible referring expression. First it checks if a single property (attribute-value pair) could be used to correctly identify the referent and rule out all of the distractors, and in that case it has finished. If not, it goes on to try all the combinations of two properties and checks if it can correctly describe the referent. Then it checks all the combinations of three properties, and so on.

In Figure 3.1 we have a series of objects of different shapes, sizes and colours. We will consider the attributes "type" (in this case the type will be the shape of the object), "colour" and "size". The types are cube, sphere, cylinder and pyramid, the colours are blue, green, yellow, orange, purple and pink, and the sizes are large and small. If the referent is item 7, the algorithm would first try to describe it by using only one attribute-value pair. As we have already explained, the attribute "type" must always be mentioned at the beginning of a description. As item 7 is the only pyramid in the scene, the referring expression generated by this algorithm would be

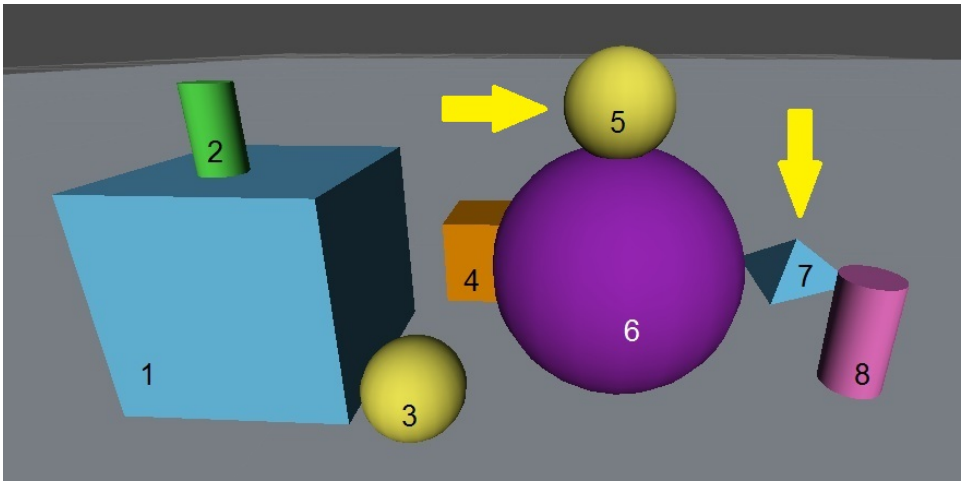

Figure 3.1: First set of coloured objects for the algorithm

"The pyramid". This would effectively rule out all distractors and distinguish the referent from the rest of the items.

Now say that the referent is item number 5. First the algorithm would attempt to use only one attribute-value pair, so it would start by generating *"The sphere"*. This would rule out items 1, 2, 4, 7 and 8, but we would still have two distractors left, numbers 3 and 6. The algorithm would now go on to use a combination of two attributes. *"The yellow sphere"* would rule out item 6, but there would still be one distractor left, number 3. *"The small sphere"* would also have the same problem. As it is impossible to find a description for the item by using two attributes, it would now try to use three. *"The small, yellow sphere"* would still describe items 3 and 5 because they are identical, and since it has run out of attributes to try, the Full Brevity Algorithm would not be able to generate a distinguishing description in this example.

This approach has a very high complexity (NP hard) and also it does not result in realistic referring expressions. As engineers we could be tempted to think that if we can describe an entity in the most efficient way possible and using the minimum amount of words, we have succeeded. But fortunately not everybody is an engineer, and people's brains do not usually follow algorithms when they talk. It has been proved by psycholinguistic research (Olson, 1970; Sonnenschein, 1984; Pechmann, 1989; Lane et al., 2006) that people usually give descriptions that have more information than is strictly necessary. It is a lot easier to quickly find somebody in a crowd if the hearer is given, for instance, information about the colour of their clothes, their hair or what area they are in, even if that information does not rule out

anybody else.

3.3.2. Greedy Heuristic Algorithm

The Greedy Heuristic Algorithm (Dale, 1989a; Dale, 1989b) is an approximation of the previous one. It starts by checking which property of the referent rules out the most distractors, and it adds it to the referring expression. It then recalculates the contrast set, eliminating any of the distractors ruled out in this last iteration. It keeps doing this incrementally until there are no distractors left and it finds a distinguishing description. This way it takes into account the context which the referent is in.

There is no backtracking in this algorithm. This means that once it has added an attribute to the referring expression it will not remove it, even if this means the description is not minimal, and even if that attribute later turns out to be unnecessary. We see this as a good thing, as it makes it more realistic. It also makes the algorithm much more efficient than the full brevity algorithm.

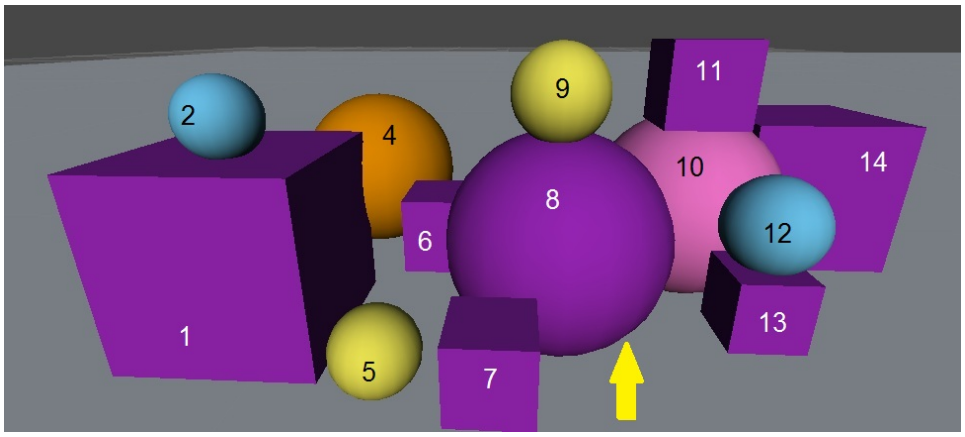

Figure 3.2: Second set of coloured objects for the algorithm

Let us say we want to describe item 8 in Figure 3.2. First of all the algorithm would choose the "type" as the first attribute, so it would generate *"The sphere"*. This rules out all the cubes, and we are left with distractors 2, 4, 5, 9, 10 and 12. If the order in which the algorithm checks the attributes is type > size > colour, it would then count how many distractors the size would rule out (four) and then check to see if any of the other attributes rules out more than that. By choosing to describe the colour, the algorithm rules out six distractors, and since there are no other attributes to check, it will include the colour in the description, which is now *"The purple sphere"*.

Since this has ruled out every distractor, it is a distinguishing description.

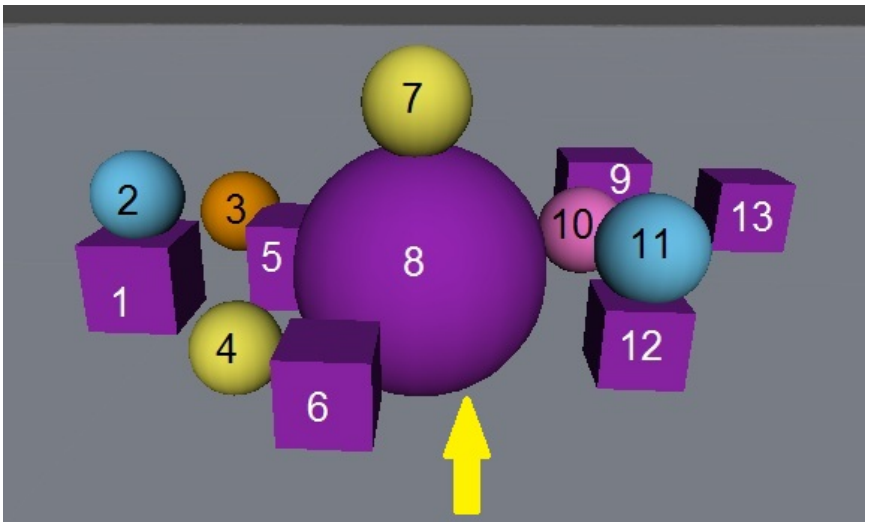

Figure 3.3: Third set of coloured objects for the algorithm

In order to describe item 8 in Figure 3.3 the process would start the same way. First the algorithm would include the "type", so the description would be *"The sphere"* and this would leave items 2, 3, 4, 7, 10 and 11 as distractors. If in this case the order to check the attributes is type > colour > size, it would check how many items the value of the colour would rule out (six). Since it rules out all of the distractors, it would generate the distinguishing description *"The purple sphere"*. But in this case, we have a lot of purple items in the scene (all of the cubes) and item number 8 is the only large object. Even though the referring expression given by the algorithm would work, it might have been easier for the hearer to identify the referent as *"The large sphere"*.

3.3.3. Incremental Algorithm

The Incremental Algorithm (Reiter and Dale, 1992; Reiter and Dale, 1995) has been one of the most influential algorithms so far. As its name suggests, it builds the referring expression incrementally, like the greedy heuristic. The difference lies in the way it chooses the order of the attributes it includes. Experimental work (Pechmann, 1989) has shown that people often prefer certain attributes over others. Sometimes the referent has a feature which would automatically discard all the distractors, but if this feature were, for instance, the colour of their shoelaces, it is very unlikely that anybody would mention that (unless maybe they were bright yellow). People tend to notice things like the colour of the referent's top, the colour

of their hair, or what posture they are in.

The incremental algorithm must have a list of attributes in the order in which they are preferred, and in each iteration it will check whether the next attribute-value pair rules out any of the remaining distractors. If it does, it will add it to the referring expression. It then recomputes the contrast set, eliminating the distractors that have just been ruled out. This algorithm does not backtrack either, so once it finds a property that it can add to the description, it will not remove it. Just like the previous algorithm, this makes it computationally efficient (polynomial).

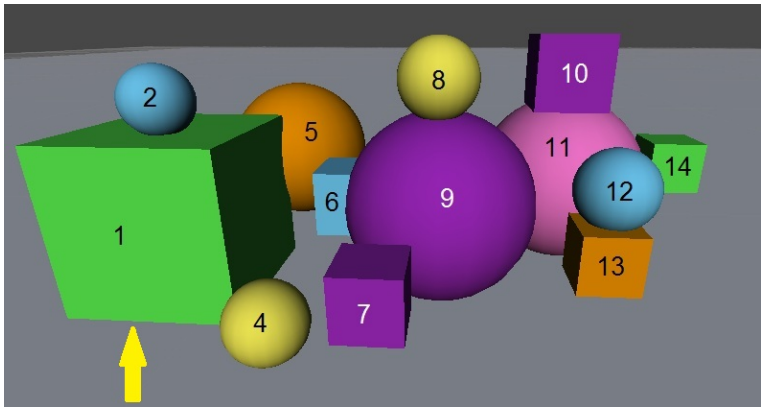

Figure 3.4: Fourth set of coloured objects for the algorithm

Let us take a look at Figure 3.4. We will consider the order of attributes type > colour > size, and try to describe item 1. The Incremental Algorithm first includes the type and generates the description (*"The cube"*), leaving us with distractors 6, 7, 10, 13 and 14. It now checks if the value of the cube's colour rules out any distractor. Since it rules out items 6, 7, 10 and 13, it will include this in the description. So far we have (*"The green cube"*). The algorithm finally checks the size of the cube. This eliminates all the remaining distractors, so the final referring expression is (*"The large, green cube"*). In this case, (*"The large cube"*) would have been enough to describe the item, but including information about the colour makes it easier to find.

3.3.4. Relational Algorithms

All of the algorithms mentioned above are a very good starting point for our work, but they are all quite limited. There are some cases in which a distinguishing description cannot be found (as we saw in the example for the Full Brevity Algorithm) and other cases in which the description generated is not what a person might normally use. When there are other

objects in the scene people often refer to them when giving a description. For instance, a person might describe the referent as *"The girl talking to the very tall man"* or *"The old man sitting by the fireplace"*. This could be done with two levels, by describing two entities in the same referring expression (as in the previous examples), on three levels (*"The hat on top of the table that is next to the door"*) or as many as the speaker wants, but the more levels we include, the harder it is to easily understand the description.

Let us go back to the first example. When trying to describe item number 5 in Figure 3.1, the Full Brevity Algorithm failed, and the Greedy Algorithm and Incremental Algorithm would also fail. This is because there are two items in the scene which are exactly the same, the only thing that tells them apart being their relation to the other objects. So in this case, item 5 could be referred to as *"The sphere on top of the purple sphere"*, or *"The small sphere on top of the big one"*, etcetera. This opens up many more possibilities and makes it possible to describe objects in a more natural way.

This type of referring expression can be very interesting. Sometimes we might have two different entities that cannot be described only by their type (for instance we might have three boys and four girls in a room), but if a relational description is used (*"The girl hugging the boy"*) it can be possible to use only the type for both entities and successfully distinguish the referent.

Some algorithms incorporate this idea (Horacek, 1996; Krahmer and Theune, 2002; Kelleher and Kruijff, 2006), but they often incorporate relations as a secondary strategy. Their algorithms only consider relations when the object's properties are not enough to describe it. Other studies, however, (Viethen and Dale, 2008) argue that people use relations even when they are not needed. So even though relations have already been explored in REG, there is still work to be done in this particular field.

3.4. Discussion

The domains in which these REG algorithms are applied are usually very basic. These scenes we have shown, with simple shapes and colours, are very good examples, but they are not complex enough to prove a real challenge. In real life situations, using just one of these algorithms is not enough to create a good, natural description of a person or an object. For our project, we intend to use a real scene, our university's canteen, and we will perform several experiments in order to base our algorithms on descriptions used by real people. We will then mix the algorithms, so we can get the best from each one of them and create an algorithm that provides the best possible description, depending on the situation.

Chapter 4

Preparing the project

First we are going to familiarise ourselves with the field we will be working on and the tools we will need for our project to work. We will define what this project will and will not be able to do, and start thinking about how we are going to implement it.

4.1. Unity 3D

In order to simulate situations as realistic as possible to test our algorithms in, we need an engine that will help us create 3D scenes.

We have chosen Unity because it allows us to very quickly create different scenes, fill them with furniture and people, and incorporate the scripts that control the application's behaviour. Another advantage this engine provides, is the possibility to run the application on many different platforms.

The first thing we have done is to download some of the free characters from the Unity Asset Store. The Asset Store provides many ready-to-use game objects, and we have found a set of three young men and three young women who fit in perfectly as students for our canteen scene. These characters are fully articulated, so it is very easy for us to place them in any posture we want. For now we have a temporary scene, which consists of three girls and three boys, and serving the purpose of objects we have a large cube (playing as the table) and a smaller one (a chair). These will be enough for us to start building our first algorithms and testing our work. Figure 4.1 is what our canteen looks like right now.

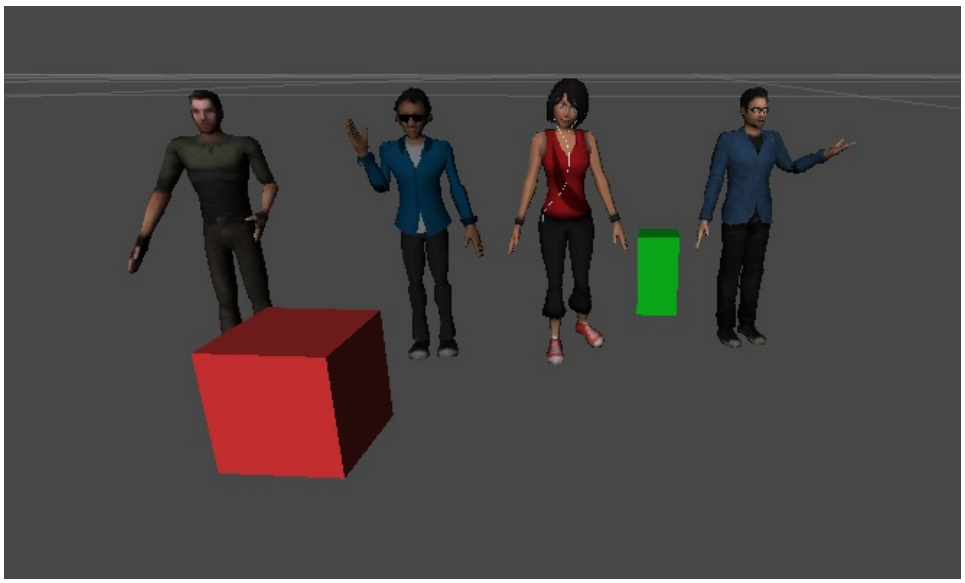

Figure 4.1: Initial scene

4.2. Design and implementation

As we will be using Unity to design the scenes and characters for the project, one of the first things we have to decide is what language we will use to implement our algorithms and the rest of the functionality. We have chosen to use C# because it is well integrated with Unity and we are already familiar with C and C++. This will allow us to create scripts that we can attach to objects and characters in the scene in order to specify their behaviour.

At this point we need to make several important design decisions. First, we need a way to store all the character information. Unity knows the coordinates in which each person will be, and also their size and orientation, but we need to store their attributes (clothes, hair colour, posture, objects they might be interacting with, etc). We have decided to store all of this information in XML files. Unity can easily read information from this type of file and add it to the game objects.

Secondly, we have to decide the general behaviour of the characters in the project. The people in each scene will be standing still, they will not be animated, and the user will also be static. This means that the camera from which the user will be observing the scene will be stationary, so the scene itself will look like a photograph. This has the disadvantage of making it harder to see people who are far away, since the user cannot zoom into them or get closer, but we will be able to start programming our algorithms

sooner if we do not need to worry about creating animations. Once we have the people placed in the scene, we need to detect if the user is clicking on a specific person. For this we will use Raycasting. Every time the user clicks on an area in the scene, Unity will send a ray from the camera to the point where the mouse has been clicked. If this ray intersects a game object, it will check if that object is a person (the objects that represent people have a tag named *"Player"* which is recognised by Unity).

Once we have reached this point, we have to include all the characters' attributes into the XML file so we can start generating descriptions for them. In order to do this correctly, the first thing we need to know is which attributes speakers normally use when describing a person. This way we will be able to not only generate precise XML files for our characters, but also to see which of their properties we should to be able to modify in order to make them different from each other. The best way to find out what real people do is to see how they phrase their descriptions when faced with this type of situation. We will create a short survey with photographs taken in our own canteen and ask people to describe some of the students, and also to identify some students for whom we provide descriptions. This will give us a very good idea of what we need to do, since the scene we are creating is the same one that appears in the photographs and the people in it should be similar. The full survey, a review of the answers and the conclusions we took from it can be found in chapter 5.

Chapter 5

First survey

5.1. Purpose of this survey

The aim of our project is not only to generate accurate and useful descriptions of the characters in the scene, but also to do it in a way that sounds natural, and as close as possible to a real human's way of speaking. In order to do this, we needed to find out how people describe each other, which details they notice and which words they choose to use when describing a situation.

We carried out this first survey before starting any of the programming, so we could hopefully start creating natural descriptions from the beginning. But this meant that we could not use our application to generate the situations for the test. We had to use photographs, and it also meant that we could not time people's responses or find out if they had any difficulty describing someone. All the pictures were taken in the canteen at our university. It is a large area that gets quite full at certain times during the day, and since it is the room we use in our project it would give us a perfect situation for the test.

The survey has two different parts. In the first part we provide a description for a certain person in the room, and the users have to guess which person it is. This shows us if the descriptions that we choose to use are accurate enough, or if they generate confusion. In the second part we ask the users to provide a description for a certain person, so we can see what each of them focuses on when describing somebody.

A total of seventy-one people answered our questions, about half of them in their twenties and most of them with a high educational level. Thirty-four (48 %) were men and thirty-seven (52 %) were women.

Figure 5.1: Gender distribution for the first survey

When analysing the answers, we are going to ignore people who did not answer or who said something that was very obviously wrong.

5.2. Links

The link to complete our survey is
`https://docs.google.com/forms/d/1UxoInunM-4qkgoZqAqfO9ogFwAH64yQXuFAhxCTeYPQ/viewform`

The link to people's basic information and their answers is
`https://docs.google.com/forms/d/1UxoInunM-4qkgoZqAqfO9ogFwAH64yQXuFAhxCTeYPQ/viewanalytics#start=publishanalytics`

5.3. Part one: Guess the person we are referring to

5.3.1 Who is the boy with the black t-shirt? (Figure 5.2)

We realise this is a trick question. There are four boys in black, but boy 3 and boy 9 are wearing black coats, so that leaves us with boy 8 (who might be wearing a t-shirt) and boy 6. We wanted to find out which of these people the users would choose in case of doubt. We have considered both 8 and 6 as a correct answer.

Figure 5.2: First part of the first survey, question 1

The results can be seen in Figure 5.3.

- Twenty-three people (32 %) answered number 9, which is wrong. He is wearing black, even though it is not a t-shirt, but out of the people wearing black he is the closest to the observer and the most visible.
- Twenty people (28 %) answered number 8, which is correct. He is in a group of people wearing dark colours (boys 9 and 10).
- Fifteen people (21 %) answered number 6, which is correct. He is in the very center of the photograph, but he is sitting right at the back, so he is harder to see.
- Five people (7 %) did not know the answer.

From these answers we concluded that people are more likely to notice someone that is closer to them, and that the colour of a person's clothes is more important that the type of the clothes. If someone is wearing a black coat they might not notice that even though the colour is right, they are not wearing a t-shirt.

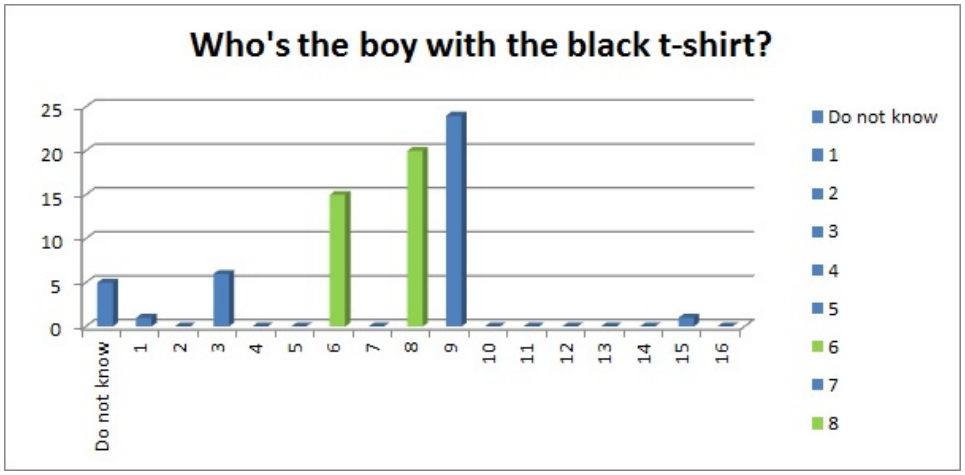

Figure 5.3: First part of the first survey, answers to question 1

5.3.2 Who is the boy leaning against the wall? (Figure 5.4)

By choosing this boy we wanted to find out if it would be easier for the users to identify a person when they are very close to an important area in the room.

The results can be seen in Figure 5.5.

- Sixty-seven people (94 %) answered number 4, which is correct. He is at the edge of the photo and he is not very visible, but he is the only one leaning on the wall.
- Two people (3 %) answered number 1, which is wrong. He is closer to the observer, and he is right at the edge of the picture, but he is not actually leaning against the wall.
- One person (1 %) answered number 5, which is wrong. He is leaning, but he is leaning on the table, not the wall. He is standing up and he is quite easy to see.

Almost everyone got this question right. Since the wall is an important part of the room, people's eyes are drawn to it quickly, making it easy for them to find the person they are looking for.

Figure 5.4: First part of the first survey, question 2

5.3.3 Who is the person sitting next to the window? (Figure 5.6)

This time, as well as choosing a person that is next to an important area of the room, we decided to pick someone who is further away from the user, to see if this had any effect on their reactions.

The results can be seen in Figure 5.7.

- Sixty-eight people (96 %) answered number 16, which is correct. Out of all the people sitting and standing next to the window, he is the closest, but he is still at the back of the photo and he is not very visible.
- Two people (3 %) answered number 14, which is wrong. He stands out the most in that group because he is standing up and wearing red

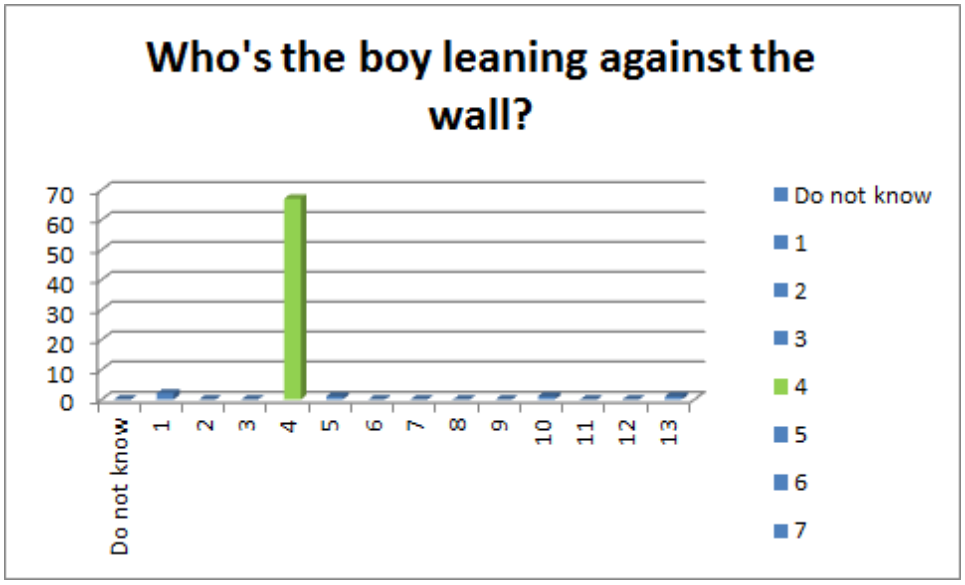

Figure 5.5: First part of the first survey, answers to question 2

clothes, and his number is also very obvious, but he is not sitting down, which is what the description said.

- One person (1 %) answered number 15, which is wrong. He is further away from the window, although he is sitting down.

Again, nearly everybody got it right. When we mention something like the wall or the window, people's eyes seem to automatically go towards that area and ignore the rest of the picture, so it is easier for them to find the person who fits the description.

Figure 5.6: First part of the first survey, question 3

5.3.4 Who is the girl with black hair? (Figure 5.8)

Here we also chose a person standing a bit further away from the user, and we decided to pick one of the only two girls with dark hair in the whole photograph.

The results can be seen in Figure 5.9.

- Forty-nine people (69 %) answered number 7, which is correct. Even though she is standing in the center of the photo, and there are only two other girls in the room, a lot of people got this one wrong. But out of all the girls, she is the one that is the furthest away from the observer.
- Sixteen people (23 %) answered number 2, which is wrong. She does have dark hair, and she is closer than girl number 7, but if we compare

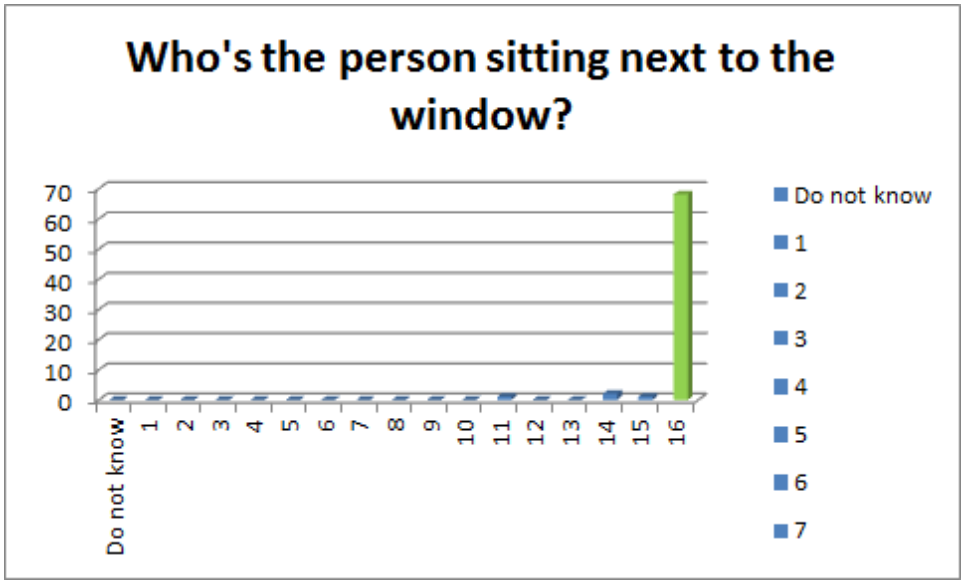

Figure 5.7: First part of the first survey, answers to question 3

their hair colour, it is obvious that the one with black hair is number 7.

- Two people (3 %) answered number 1, which is wrong. This girl is blonde, but she is at the front of the photo.
- Two people (3 %) did not know the answer.

From these answers we can see that people focus first on what they first see. If they see girl number 2 before girl 7, they will probably think she is close enough to the description and they will not look any further. For this reason it might be a good idea to provide more details than necessary when describing a person that is further away.

Figure 5.8: First part of the first survey, question 4

5.4. Part two: Describe the person we are referring to

5.4.1 Describe person number 2 (Figure 5.10)

There were many different descriptions for boy number 2, but we have detected some patterns that several people have used in their description.

Thirteen users (19 %) described him by mentioning only his posture (here we have also included references to his laptop as part of his posture), nine users (13 %) described him only by his attitude and nine more mentioned his clothes and his posture, and finally eight users (11 %) described everything about him (his clothes, his posture, his position in the room, his physical features, etc). The rest of the users gave slightly more varied descriptions, but forty-six people (66 %) mentioned his posture in some way,

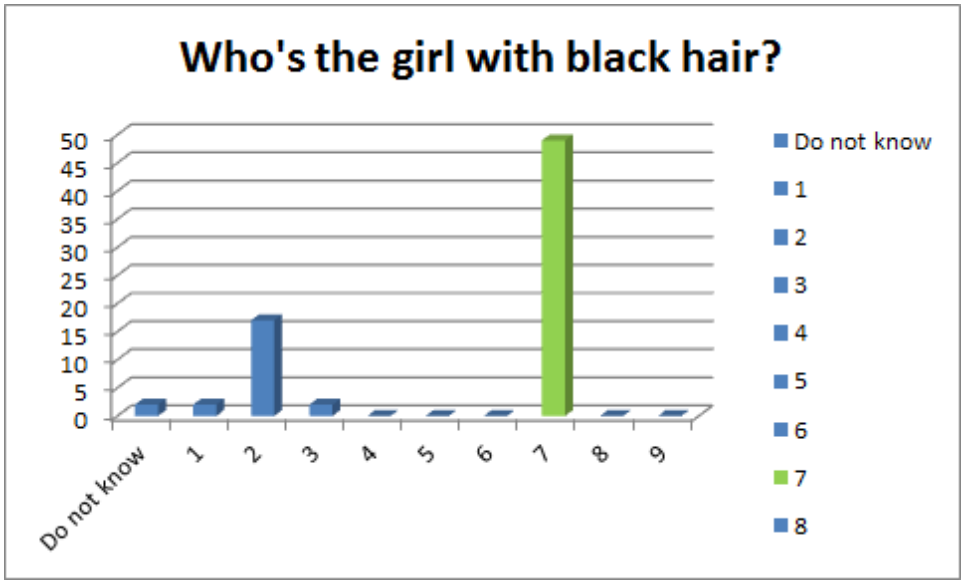

Figure 5.9: First part of the first survey, answers to question 4

and twenty-five of them (36 %) mentioned his clothes.

We can conclude that in this case, since the referent is in a very particular posture (hands crossed beneath his chin and looking at his laptop), the users have a tendency to include this as the main part of their description (or in some cases, the whole description consists in his posture). There is only one other person in the photograph with a laptop, and nobody else visible with their hands under their chin. For this reason his posture stands out and is a very descriptive feature.

Figure 5.10: Second part of the first survey, question 1

5.4.2 Describe person number 8 (Figure 5.11)

Ten users (14 %) described woman 8 only by her clothes, seven (10 %) mentioned her attitude, and six people (9 %) used only her profession (waitress). Also in this case we should note that her clothes (she is wearing an apron) are related to her profession, so that is what stood out the most to everybody. Overall, forty-one people (59 %) mentioned her clothes, and twenty-nine people (41 %) mentioned her profession.

When someone is clearly recognisable by their job, in this case a waitress but it could also be a bartender or a person who is cleaning, this can be descriptive enough and we might not need to mention anything else. This could also be applied to the person's *type*. In this case, woman number eight's type would be *"waitress"*. In our project we are not going to include any people with an identifiable profession, so this specific example will not

Figure 5.11: Second part of the first survey, question 2

apply to our work, but we can include people of different races. For instance, if we had a black girl, her type would be *"black girl"*. This way, if she is the only coloured person in the room, her type should suffice to create a distinguishing description for her.

5.4.3 Describe person number 12 (Figure 5.12)

Figure 5.12: Second part of the first survey, question 3

In this case boy number 12 is barely visible, even his face is partly hidden. Six people (8 %) gave an exhaustive description of everything they could see, but a lot of people described him by his clothes (thirty-seven people, 53 %) even though there are other boys in the picture, and in his same area, who are wearing clothes of a similar description (white t-shirt with dark details).

Even when there are several people in a scene wearing similar clothes, people often tend to include information about those clothes in their description.

5.4.4 Describe person number 4 (Figure 5.13)

Figure 5.13: Second part of the first survey, question 4

For boy number 4, fifteen people (21 %) described him as the boy with the red shirt, and did not mention anything else, even though there is another boy (number 10) that could also fit in that description. A few people also noticed his posture (seventeen people, 24 %) and the fact that he is in a group of people.

This reinforces what we concluded earlier in the first part of the test: when people see someone who fits a description, they do not look any further to see if maybe that description also applies to someone else.

5.4.5 Describe person number 10 (Figure 5.14)

Figure 5.14: Second part of the first survey, question 5

We chose boy number 10 because he is also sitting with a group of friends and wearing a red shirt, so his description might be very similar to boy 4. This time, a lot of people (twenty-two, 31 %) described his posture as well as his clothes, and said that he is talking to the boy next to him. Even though at this point everyone should be aware that there are two boys in the picture with a red top, ten people (14 %) still described only his clothes, but here they mentioned that his top has long sleeves.

It seems that the clothes a person is wearing are one of the most important attributes that users mention in their descriptions. Sometimes even when the colour of their clothes alone is not enough to distinguish a person, if it stands out enough, users tend to sometimes mention only that.

5.4.6 Describe person number 3 (Figure 5.15)

Figure 5.15: Second part of the first survey, question 6

We chose boy number 3 because his face is not visible, the colour of his clothes does not stand out, and there seems to be nothing particularly eye-catching about him. Here almost everybody (fifty-one people, 73 %) described his posture (he is sitting facing away from the observer), and most mentioned that he is sitting next to a girl. Some even described the girl's clothes, because they stand out more than his. A lot of people (thirty-six, 51 %) also used his clothes to describe him.

Here we can see that when a person does not stand out very much, people tend to notice something nearby that stands out more (in this case the girl he is sitting with, but it could also be a window, a door or an object like a laptop).

5.5. Conclusions

Based on the results we have obtained, we can see that rather than giving the shortest and most efficient description possible, people often give more information than is needed. This makes it easier for us to find the right person quickly. Therefore, our algorithm should not focus on generating the minimum information required to identify the character; it should generate the information that people think is the most descriptive.

Our test subjects have mostly focused on the colour of the clothes, posture and immediate surroundings. When someone is interacting with an object (for instance a laptop) or another person, this can be very important information.

A very curious thing that we noticed is that people tend to notice beards and glasses a lot. Even in pictures where the person did not actually have glasses (boy number 2 from question 5) if people think that he is wearing them, they will say it. When someone has a feature that stands out, such as glasses or a beard or moustache we should always mention it. If the character does not really stand out, rather than giving an exhaustive description we might want to describe the person next to them.

We found it surprising that a lot of people chose to describe someone based on their attitude or the personality they think they have. But since this is very subjective, we will not be using it to generate our referring expressions.

Lastly, we should always try to use important areas in the room, such as a window or a door, to describe the character if they are near enough to it.

Chapter 6

First Iteration

In this iteration we will start building algorithms based on the information we collected from the first survey. We will also add more characters and some furniture to the scene to improve it and make it more realistic. Lastly we will have to modify the appearance of each character so that it matches the information described in the XML file.

6.1. Introduction

Our first survey allowed us to better understand the details that people notice when describing another person. We have created a list of the attributes we will include with each character, the first ones being the ones that people tend to focus on the most.

By order of importance:

1. type (always included in the referring expression)
2. top colour
3. posture
4. beard (if the person has a beard)
5. hair colour
6. top type
7. hair length
8. bottom colour
9. bottom type

As we have already explained, the *type* attribute should always be included. This is the noun that defines the referent, and everybody uses it unconsciously when describing a person. The two properties that the users mentioned the most were the colour of the person's clothes and their posture. Most users tend to notice the colour, because it usually stands out more in the scene, but the posture in most cases was also very important. As we concluded in the first survey, people often notice if the referent has a beard, so it should also be included if the person has one. The rest of the attributes are less important. In our scenes most of the people have dark hair so information about its colour is not always useful, and hair length is not a property that stands out at first glance. The type of the top seems to be less important than its colour, and the type and colour of the bottom is usually quite hard to see, especially if the referent is sitting down.

This leaves us with the following structure for a character's information, which we have included in a DTD file:

```
<?xml version="1.0" encoding="UTF-8"?>

<!ELEMENT people (person*)>
<!ELEMENT person (type, name, sex, height, hair, beard,
    clothes, posture, realposture, object*, special)>

<!ELEMENT hair (length, colour)>
<!ELEMENT clothes (top, bottom)>

<!ELEMENT top (type, colour)>
<!ELEMENT bottom (type, colour)>


<!ELEMENT type (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT sex (#PCDATA)>
<!ELEMENT height (#PCDATA)>
<!ELEMENT beard (#PCDATA)>
<!ELEMENT posture (#PCDATA)>
<!ELEMENT realposture (#PCDATA)>
<!ELEMENT object (#PCDATA)>
<!ELEMENT special (#PCDATA)>

<!ELEMENT length (#PCDATA)>
<!ELEMENT colour (#PCDATA)>
```

This DTD checks that the structure of the XML file correct. Here is an example of a character's information in the XML file:

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE people SYSTEM "peopleDTD.dtd">

<people>
    <person>
      <type>boy</type>
      <name>chico1</name>
      <sex>male</sex>
      <height>165</height>
      <hair>
          <length>short</length>
          <colour>black</colour>
      </hair>
      <beard>no</beard>
      <clothes>
          <top>
              <type>sweater</type>
              <colour>black</colour>
          </top>
          <bottom>
              <type>jeans</type>
              <colour>dark</colour>
          </bottom>
      </clothes>
      <posture>standing</posture>
      <object>phone</object>
      <object>watch</object>
      <special>braid</special>
  </person>

    ...
    ...

</people>
```

The *name* of the character is the name the Game Object will have when it is created in Unity. The *objects* are optional attributes that represent objects the character is carrying or interacting with, and the *special* attributes are also optional and represent special traits that make the character stand out, such as glasses.

6.2. Basic algorithms

Now that we have included all the above mentioned attributes in the characters' descriptions, we are ready to start creating our algorithms. We have not used the Full Brevity Algorithm described in chapter 3 because the referring expressions it generates do not imitate natural language. The

first algorithm we will build is the most simple, the Exhaustive Description Algorithm.

6.2.1. Exhaustive Description

The **Exhaustive Algorithm** generates a full physical description of the referent. It makes use of all the attributes that describe them physically, and it generates a description that imitates natural language.

The algorithm's behaviour is very simple. It reads through the attributes in the XML file where the character's description can be found, and it stores them in local variables. It then strings these variables together in order to create a full sentence. The order we have used for the attributes is not the one described above, by order of importance, but we have chosen the order to fit in with the way people usually create their sentences. Here are two examples of descriptions that could be generated for Figure 6.1:

Figure 6.1: Example with two characters

1. The boy with short black hair and a beard, with the green shirt and gray trousers.

2. The girl with short brown hair, with the pink top and blue trousers.

The information stored in the attributes depends completely on the programmer. In this case for example, the girl's hair is considered short, because in general a girl described as having long hair might have her hair down to her waist. On the other hand if the boy had shoulder length hair like her, he could be described as the boy with long hair, since it is more unusual for men to have longer hair. The colour of the boy's trousers is also not very clear, it could be considered gray, green or even brown. This kind of information of course is purely subjective and can be easily modified in the XML file.

6.2.2. Relational Algorithms: Nearby People and Objects

We will now use the Exhaustive Description algorithm as a base for our next two algorithms. As we mentioned in chapter 3, relational algorithms have a lot to offer and we wish to further explore them. From our first survey we concluded that people tend to refer to important areas of the room if the referent is close enough to them, and sometimes they refer to nearby people as well, especially when the referent does not stand out very much. For this reason we have decided to create two basic relational algorithms: the Nearby Objects algorithm and the Nearby People algorithm.

These two algorithms are both quite similar. Instead of describing the actual referent, they find the closest object or the closest person, and describe them. We have considered the referent to be *next to* an object or person if they are 0.5 meters or less away form them, and they are *near* an object or person if they are between 0.5 and 1.5 meters away. If there are not any objects or people nearby, the algorithms will simply provide the Exhaustive Description for the referent and there will be no relations. This is a first approach to these algorithms, since it may not be useful to describe the person or object that is closest to them, but the one that stands out the most. We will modify both these algorithms further on, if necessary.

When generating the **Nearby people** description, we start off by including the *type* of the referent. This will be the only information that refers to them. Next we check to see if the nearby person chosen by the algorithm is next to the referent or near them, and we include this in the description. Lastly we generate an exhaustive description for that nearby person.

The **Nearby objects** description is generated in a very similar way. First the *type* of the referent is included, then we check to see if the object is next to the referent or near them and include it in the description, and then we include the *type* of the objects. We have considered objects to have only their *type* as attribute, because we will only be including important areas of the room as objects. Since the canteen is full of tables and chairs,

it does not make sense to include these as objects, we will only be including the window, the bar and the three columns. The window and the bar need no further description, so they only need their *type*, and since we only have three columns, we have considered their types to be *"first column"*, *"second column"* and *"third column"*. The first column is the one closest to the observer, and the third one is the one that is the furthest away. A photograph of the canteen can be seen in Figure 6.3.

As we mentioned in the State of the Art, these relational algorithms can have several levels. A description like *"The girl next to the window"* has two levels, *"The cat sleeping in the basket on top of the table"* has three levels, and so on. We have decided to include only two levels, because we find the descriptions clearer and easier to understand this way. We have been careful not to allow loops by making sure that the nearby person gets described by the Exhaustive algorithm, and not by one of the relational ones. Otherwise we might end up with descriptions like *"The boy next to the girl next to the boy next to the girl..."*

Figure 6.2: Example with two characters and a window

In Figure 6.2 we have a blue shape temporarily playing as the window. In this scene the boy could be described as:

- Nearby People algorithm: The boy next to the girl with short brown hair, with the pink top and blue trousers.

- Nearby Objects algorithm: The boy near the window.

Of course these basic relational algorithms are a first approach to what we want to do, because they do not do a very good job of distinguishing the referent. If there were five boys near the window, the Nearby Objects algorithm would still generate the same description and it would not be correct.

6.3. More advanced algorithms

Now that we have some basic algorithms to work on, it is time to build two of the algorithms mentioned in the State of the Art.

6.3.1. Incremental Algorithm

First we will create the **Incremental Algorithm**. This one considers the priority of each attribute mentioned in section 6.1 in order to create more realistic descriptions.

We start off with a list of distractors (at first all of the people except for the referent), an empty description string, and the list of attributes it will consider in order of priority. We also have a variable for each attribute, initialised as an empty string (“”). This algorithm starts checking all of the referent's attributes one by one. The only one it always includes no matter what is the *type*, so it saves it in its variable. Now it checks to see if the *type* has ruled out any of the distractors, and if so, it will remove those people from the list of distractors. As its name suggests, the algorithm works incrementally. During each iteration it will pick the next attribute in the priority list and check if this rules out a distractor. As long as one person is ruled out, the algorithm considers this a success and it will save that attribute in its variable. It then deletes the ruled out people from the list of distractors and continues on to the next iteration. It will keep doing this until there are no distractors left (it has successfully generated a good referring expression) or until it runs out of attributes to check (in this case it is not possible to generate a distinguishing referring expression with this algorithm). As we explained earlier, the Incremental Algorithm does not remove an attribute once it has included it. This might make the referring expression longer than necessary, but it makes it more similar to descriptions created by people.

This algorithm provides descriptions that sound closer to a real person's way of speaking. People tend to consider some properties more important than others no matter what the situation looks like, and so they mention

these properties first. Of course the actual attributes considered to be important vary from one person to the next, and sometimes the same person will choose them differently even for the same situation. It is impossible to create an algorithm that suits everybody and that everyone agrees with, but we have come as close as we possibly can to this, since the order of attributes we have chosen is based on the answers of the seventy-one people that completed our first survey.

6.3.2. Greedy Algorithm

The other algorithm we will create is the **Greedy Algorithm**. Like the Incremental Algorithm, this one will generate more realistic descriptions than the basic one. This algorithm will choose the attributes it mentions based on the situation. It will create different referring expressions depending on the people in the scene, always choosing the attribute that is the most distinguishing for the referent with the remaining distractors.

The Greedy Algorithm also starts with a list of distractors, an empty description string, a list of attributes ordered by their priority and an empty variable for each attribute. Just like the other one, it will first include the *type* attribute and delete all the distractors that have been ruled out. Now for each of the remaining attributes in the attribute list, it checks how many distractors that specific one would rule out. After reviewing all the attributes it chooses the one that rules out the most distractors. If two or more attributes rule out the same amount of people, the one with the highest priority is used. Now the algorithm updates the list of distractors, deleting the ones that have been ruled out, and the process will continue with the remaining attributes. When there are no more distractors left, or when it has used all the attributes (like the Incremental Algorithm, it might not be able to successfully generate a distinguishing referring expression) it finishes. Once again, this algorithm does not remove an attribute once it has been included in the description.

The Greedy Algorithm is good for generating shorter referring expressions that still imitate human descriptions. Since it chooses the most distinguishing attributes each time, it uses properties that are more eye-catching for the observer, so the referent can be found faster. On the other hand, these properties can sometimes be hard to see. For instance if a person with bright yellow trousers is sitting down, it might be hard to see the colour of their trousers even if this is the referent's most distinguishing property.

6.4. Merging the algorithms together

Now that we have a few more algorithms, it is time to mix them together to generate more sophisticated referring expressions. We will combine our relational algorithms with the Incremental and Greedy algorithms. So far the Nearby People Algorithm relies on the Exhaustive Algorithm to generate the description for the nearby person, and the referent in both relational algorithms is only described by his or her type.

We will modify the Nearby Objects Algorithm so that the referent's description can also be generated using the Incremental Algorithm or the Greedy one. This way we now have three different versions of this relational algorithm.

The Nearby People Algorithm will be modified so that the nearby person can also be described with any of these two new algorithms. For the referent, instead of using just their type, we will use the Greedy Algorithm, since it provides more information. If we used only the referent's type, the algorithm would not be able to generate a distinguishing description for the referent. It would generate descriptions like *"The boy next to the girl in the red dress"*, and there could be many boys to whom that description would apply.

At this point our algorithms are:

- Exhaustive Algorithm
- Incremental Algorithm
- Greedy Algorithm
- Nearby Objects Algorithm with Exhaustive
- Nearby Objects Algorithm with Incremental
- Nearby Objects Algorithm with Greedy
- Nearby People Algorithm with Exhaustive
- Nearby People Algorithm with Incremental
- Nearby People Algorithm with Greedy

6.5. Creating the scenes

With all the code ready to run a second test on users, we need to create some scenes in Unity to test our algorithms on. In the first survey we took

photographs of our canteen, so in preparation for the next survey we will recreate some of those pictures.

Figure 6.3: 3D model of the canteen

We have a 3D model of our canteen (see Figure 6.3), very similar to the real one. Our next step will be to create characters to populate the scene. We will use the three men and three women that we downloaded from the Asset Store and modify their appearance with Photoshop. We can change their skin colour, hair colour, the colour of their clothes, and slightly modify their clothes to turn short sleeves into long sleeves or viceversa, or full trousers into cropped trousers. We can also add beards to some characters, and glasses to others. Each character's texture has to be modified individually, and the new texture should be included in the same folder where the character is, so it can be loaded correctly.

The characters' postures are easy to modify inside Unity, but not so much from the scripts. Inside Unity, the character's articulations can be rotated and dragged, and any posture can be created in a matter of minutes. From a script, on the other hand, the exact angle of each articulation must be set. We first created all three scenes inside Unity, placing each character in a particular spot and giving them their own posture. This means that for each character created (there are more or less twenty people per scene) there is an individual posture, and a set of coordinates that represent that character's position in the scene. Since we are going to generate several scenes with each execution, we load each person's posture once the character has been instantiated, and this must be done with a script. Each character

is assigned a *Posture* component, represented by the script *Posture.cs*. This script contains a set of very specific postures, that go by the name of *chico1*, *chico2*, *chico3*, etc. Each of them holds the exact angles of each articulation for a specific posture and a set of coordinates. An example of code for a particular posture can be seen in Figure 6.4.

```
switch(type){
    case "chico1":
        person.transform.localPosition = (new Vector3
                ((float)35.41526,(float)0.768602,
                (float)1.049184));
        person.transform.localEulerAngles = (new Vector3
                ((float)0,(float)319.5383,(float)0));
        person.transform.localScale = (new Vector3
                ((float)0.7,(float)0.7,(float)0.7));
        leftUpLeg.Rotate((float)0,(float)0,(float)0);
        leftLeg.Rotate((float)0,(float)0,(float)0);
        rightUpLeg.Rotate((float)0,(float)0,(float)0);
        rightLeg.Rotate((float)0,(float)0,(float)0);
        leftShoulder.Rotate((float)0,(float)0,(float)0);
        leftArm.Rotate((float)343.4716,(float)21.81518,
                (float)78.52637);
        leftForeArm.Rotate((float)0,(float)0,(float)0);
        leftHand.Rotate((float)0,(float)0,(float)0);
        rightShoulder.Rotate((float)0,(float)0,(float)0);
        rightArm.Rotate((float)346.4534,(float)301.4185,
                (float)301.5587);
        rightForeArm.Rotate((float)0,(float)0,(float)0);
        rightHand.Rotate((float)0,(float)0,(float)0);
        spine.Rotate((float)0,(float)0,(float)0);
        spine1.Rotate((float)0,(float)0,(float)0);
        spine2.Rotate((float)0,(float)0,(float)0);
        hips.Rotate((float)0,(float)0,(float)0);
        neck.Rotate((float)7.586344,(float)323.2532,
                (float)6.787269);
        break;
```

Figure 6.4: Fragment of code

Once all these different postures have been created inside the script, they have to be assigned to each character. For that purpose, we have modified the XML file and added an extra attribute, *realPosture*. It now looks like this:

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE people SYSTEM "peopleDTD.dtd">

<people>
    <person>
      <type>boy</type>
      <name>chico1</name>
      <sex>male</sex>
      <height>165</height>
      <hair>
         <length>short</length>
         <colour>black</colour>
      </hair>
      <beard>no</beard>
      <clothes>
         <top>
            <type>sweater</type>
            <colour>black</colour>
         </top>
         <bottom>
            <type>jeans</type>
            <colour>dark</colour>
         </bottom>
      </clothes>
      <posture>chico1</posture>
      <realposture>standing</realposture>
      <object>phone</object>
      <object>watch</object>
      <special>braid</special>
   </person>

    ...
    ...

</people>
```

The *posture* attribute represents the technical posture set inside the *Posture.cs* script. The *realposture* attribute, on the other hand, represents the posture as it would be called in real life. This could be *"standing"*, *"sitting"*, *"leaning on a table"*, etc. When each character is created, its posture is modified to look like the one listed in its attributes. Boy 1 would get posture *chico1*, boy 2 would get posture *chico2*, and so on.

Since we have over sixty characters, it would have been very tedious to write all the angles for all the articulations of each posture manually. As we mentioned above, we first created the scene in Unity and manually placed all of the characters in their place. We wrote a script that would read each

of the articulations' angle for each of the characters, and write the fragment of code that could set that angle. For instance given a particular angle for a person's right arm, this script can generate the code

```
rightArm.Rotate((float)346.45,(float)301.41,(float)301.55);
```

The same applies to the person's coordinates and their scale. The *scale* is the size of the character, we have set all the scales to be approximately 0.7 so they fit in with the size of the canteen.

This way we have set all the postures to imitate the scenes from the photographs of the first survey. We will not recreate the scenes exactly, because we have limited time to complete this project, so we have included about twenty people in each scene and we have not used any objects (such as laptops or food). Although we have three *scenes* when we run the application, inside Unity there is only one scene. Every time a new person has to be described, the old characters are removed and the new characters are created in their place. The resulting situations are figures 6.5, 6.6 and 6.7.

Figure 6.5: Scene 1

In this version of the application, once all our characters were in the scenes, we needed to generate the description for a particular character and print it on the screen after the user clicks on them. As we have mentioned before, we use Raycasting for this. In our main script, *World.cs*, there is an Update method which is called once per frame. If a mouse click is detected, this method is the one responsible for sending a ray from the character to the coordinates where the click has been detected. It checks whether that ray has hit an object, and if it has, it then checks to see if the object is a character (if it is a character it will have a *"Player"* tag). In that case, it calls the method responsible for generating the character's description. Here we encountered a minor problem. When our application is launched, it first creates all the characters and places them in the scene, and it then sets their postures and adds a Mesh Collider that allows the characters to be

Figure 6.6: Scene 2

Figure 6.7: Scene 3

clickable. This Mesh Collider cannot be articulated like the game objects, so it stays in the default posture that the characters were downloaded with, as can be seen in Figure 6.8.

This means that when the user clicks on the character, they can only click on the area covered by the collider. If they were to click on this character's arm or head for example, the click would not be detected. We have solved this by inserting several colliders of different shapes on some of the characters' articulations, as can be seen in Figure 6.9. For example we have capsules on their arms and legs, or a cube on their torso. This has the disadvantage of offering less precision when clicking on the character, but for our purpose it works well.

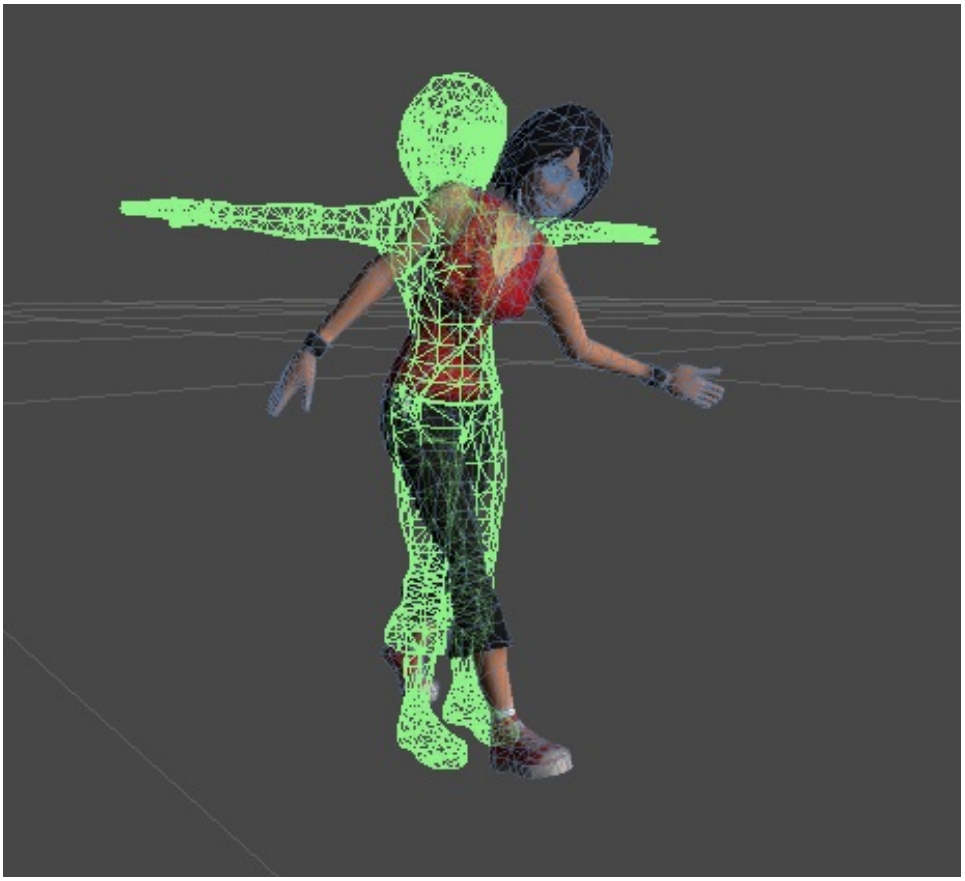

Figure 6.8: Difference between Mesh Collider and Game Object

6.6. Generating the referring expressions

Once we can detect collisions with the characters, the only thing left to do is to generate and print their description. In the Update method inside the *World.cs* script, the method that generates the referring expression is called when it detects that a character has been clicked on. This method can be any of the algorithms that we mentioned in section 6.4 (Exhaustive, Nearby Objects, Incremental with Nearby People, etc.). To change the algorithm that is used, the other algorithms must be commented(//).

For this part, there is a script that plays an important role, the *Counter.cs* script. This script will hold a reference to all the characters in the scene that have a particular value for a specific attribute. It contains a *name* string, which will represent both the attribute and its value. For example the *name* could be *"colourTop blue"*. It contains the integer *counter* which counts how many times that particular value for the attribute appears. It

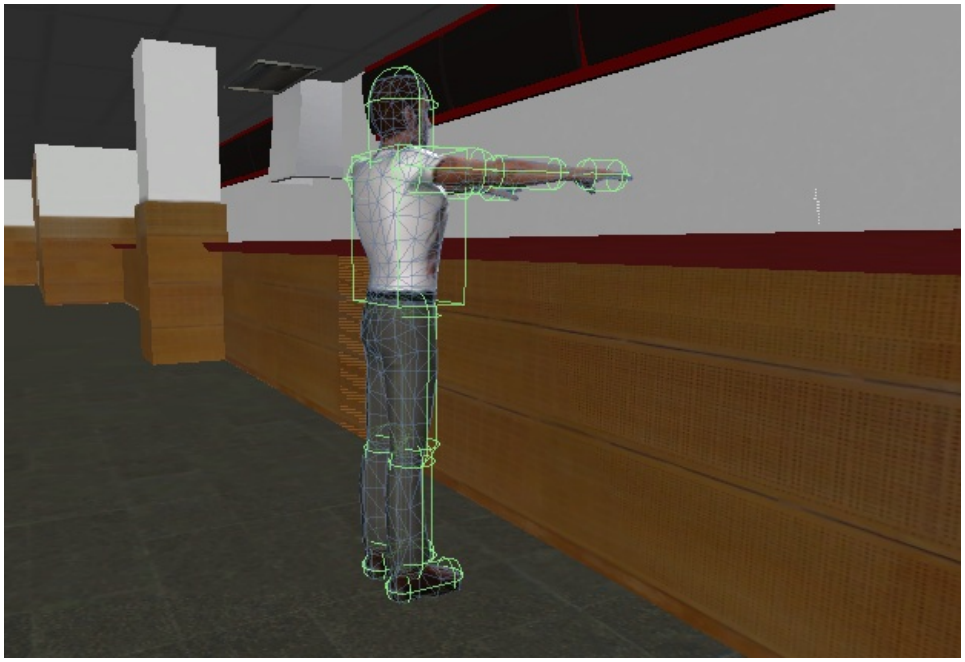

Figure 6.9: Different shaped colliders on a character

contains a string called *type* which is unused but we have kept for future work. This will store the value *ordinary* or *special* depending on the type of the attribute. Lastly, it contains a list of integers, *idList*. This stores the ID of the characters that have that same value for that attribute (in this example, it would store the ID of the characters that have a blue coloured top).

Every time a character is created with a new attribute that does not have a Counter, a new Counter is created. When a character is created and there is already a Counter with that value for an attribute, it will simply increment by one the number of people that have this value (*counter*) and add the character's ID to the list. This script will allow us to use the algorithms that consider the rest of the scene when choosing the order of the attributes. Most of the algorithms receive as input a list of counters for the whole scene, *allPeopleData*.

Some algorithms also receive an integer called *farthest*. This represents the distance from the observer in which the character that is the furthest away is standing. We use it to decide if a character is near or far (we consider everybody from halfway in between the camera and *farthest* to be near, and the other half to be far. The algorithms that consider nearby objects receive a list of all the objects in the scene (windows, pillars and bar).

A full explanation of all the algorithms can be found in sections 6.2 and 6.3.

6.7. Conclusions

Now that the application works as expected, we are ready to run a second survey. We need to find out which of these algorithms people prefer, and find out if different algorithms work better or worse in different situations and why. Our goal is to create a meta-algorithm that combines the others depending on the situation to create the most appropriate referring expression.

Chapter 7

Second survey

7.1. Purpose of this survey

Thanks to the first survey, we were able to create specific algorithms to generate descriptions, that imitated natural language and were based on real life situations. The next step is to make sure that the referring expressions generated by these algorithms work well with our users. For this purpose we have created a second survey which will collect the descriptions they use to describe specific characters in the scene, as well as ask them to evaluate how good they think the descriptions generated by our application are.

In this survey, instead of photographs, we have used scenes and characters created in Unity. This way we will also be able to appreciate the differences between the descriptions given for a photograph of a real scene, and a scene developed in a virtual 3D environment.

Lastly, with the answers and conclusions we reach from the survey, we will be able to improve the application. We will create a meta-algorithm that chooses the best possible combination of our algorithms depending on the situation and the rest of the people in the scene. This way, we will be able to identify the referent in the smallest possible amount of time.

A total of fifty-two people answered our survey before we started analysing the results. 54 % were women and 46 % were men. This time we have not asked our users for their level of education, since in the first survey it did not offer us any insight, but we have asked them to include their age. Most of them (67 %) are between eighteen and thirty years old, 17 % are between thirty and forty years old, 4 % are under eighteen, and 12 % are over forty. This information can be seen in Figures 7.1 and 7.2.

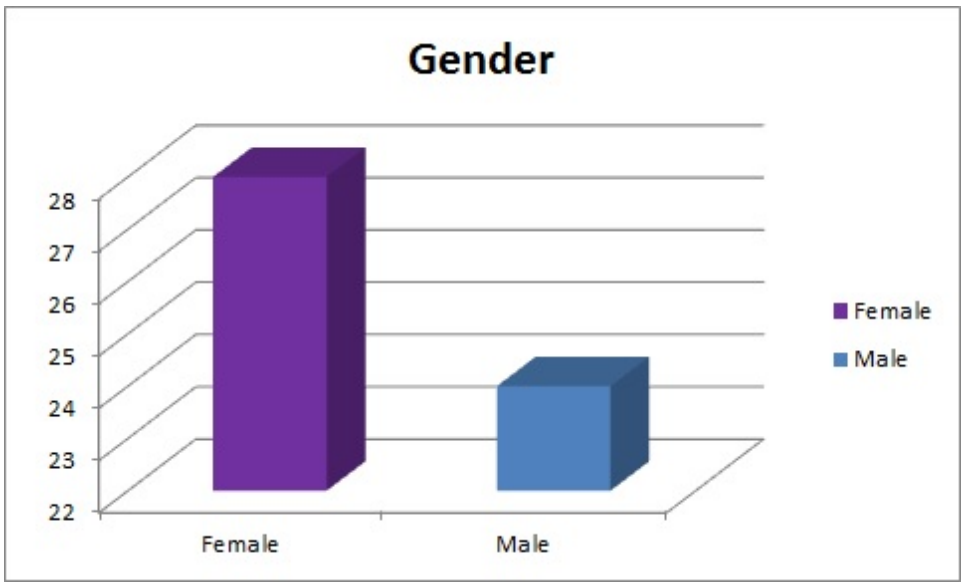

Figure 7.1: Gender distribution for the second survey

The disposition of the questions and the structure of the survey has been carefully planned so it does not influence the users' opinions. We wanted them to offer their own descriptions first, before reading and judging the descriptions generated by the algorithms. We have also considered the effort and amount of time that they will have to spend on the survey, so they will not be tempted to leave it unfinished and we can get as many answers as possible. The more results we obtain, the better we will be able to analyse them.

When analysing the answers, we are going to ignore people who did not answer or who said something that was very obviously wrong.

7.2. Links

The link to complete our survey is
`https://docs.google.com/forms/d/1KuaivLbspIo-84CopWXGqZjrFMTvMShzJD3YPVz37NuU/viewform`

The link to people's basic information and their answers is
`https://docs.google.com/forms/d/1KuaivLbspIo-84CopWXGqZjrFMTvMShzJD3YPVz37NuU/viewanalytics`

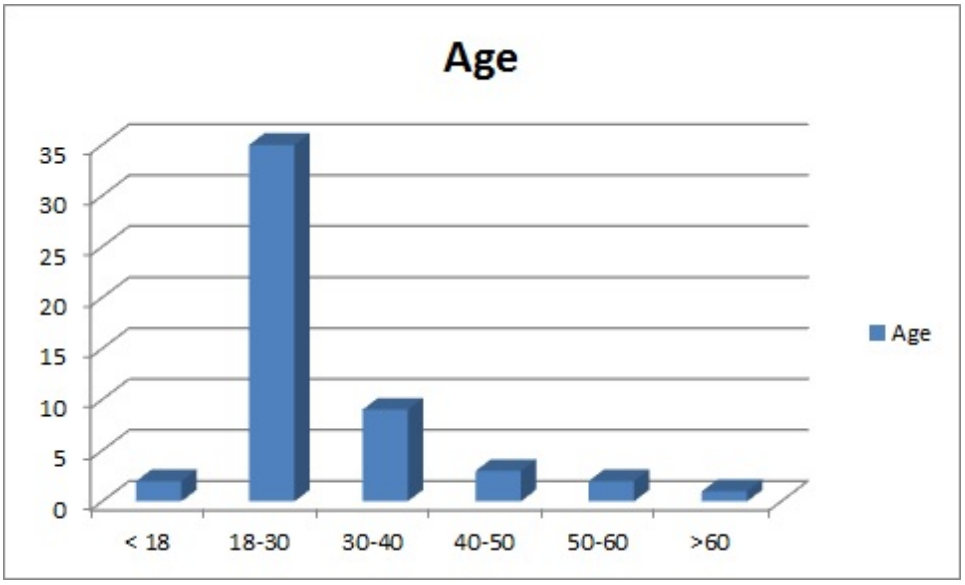

Figure 7.2: Age distribution for the second survey

7.3. Part one: Describe the person we are referring to

In this first part, we will ask the users to describe a certain person in the scene that is pointed out by an arrow.

7.3.1 Question 1: Describe the person pointed out by the arrow (Figure 7.3)

Most of the descriptions we have obtained for this photograph contain the colour of the character's top garment, his posture, and the person he has next to him. These results are very similar to the referring expression generated by our algorithms.

Figure 7.3: First part of the second survey, question 1

7.3.2 Question 2: Describe the person pointed out by the arrow (Figure 7.4)

In this question, there are eleven people (13 %) who have referred to the boy sitting next to the referent in their description, and seven people (11 %) who have mentioned the window. It seems in this case the users have preferred to describe a large important area that is close to the character and stands out.

The attributes that the users have mentioned the most are the colour of her clothes and her posture. We can see that in this question, most people thought that the visible area of skin on her upper back was part of her clothes, so instead of seeing the white top, they saw a pink top. This situation did not exist in the photograph, since the real girl's top was covering her whole back.

Figure 7.4: First part of the second survey, question 2

7.3.3 Question 3: Describe the person pointed out by the arrow (Figure 7.5)

In this question, there are twelve people (23 %) who have referred to the girl standing close to the referent (four of these people referred to her as *"The blonde"*, five people said *"The girl dressed in black"* and three people simply said *"The girl"*). This time, only four people (10 %) have mentioned the window. It seems that due to the camera's position in this image, the window is less clearly visible and cannot be seen as easily as in the previous question.

Most of the people have opted for describing the referent, using the attributes that stand out the most: his posture and his clothes.

7.4. Part two: Rate the descriptions

In this second part we have provided several different descriptions (generated by our own algorithms) and we have asked the users to rate them on a scale of one to five (one being that they dislike that description very much, three being neutral, and five being that they feel it is a very good

Figure 7.5: First part of the second survey, question 3

description). We have a total of nine algorithms so far, but we did not use all nine of them in each scene, as we felt too many descriptions would only confuse the users. For each question we chose algorithms that we wanted to compare to each other.

7.4.1 Question 1:Rate these descriptions (Figure 7.6)

The results obtained are the following:

- Greedy: *"The boy in the red sweater with dark trousers."* 67 % bad or very bad. 11 % good
- Incremental: *"The boy in the red sweater who is sitting down, he has a beard and black hair. He is far."* 22 % bad or very bad. 51 % good or very good
- Exhaustive: *" The boy with short black hair and a beard, with the red sweater and dark trousers."* 49 % bad or very bad. 22 % good or very good
- Nearby People with Exhaustive: *" The boy with short black hair and a beard, with the red sweater and dark trousers. He is next to the the boy*

Figure 7.6: Second part of the second survey, question 1

in the black and grey striped top." 4 % bad or very bad. 86 % good or very good

- Nearby People with Incremental: *"The boy in the red sweater who is sitting down, he has a beard and black hair. He is far. He is next to the boy in the black and grey striped top."* 0 % bad or very bad. 92 % good or very good

From the results (figure 7.7) we can see that the users prefer to describe the boy standing next to the referent, because he stands out more. More specifically they prefer the Incremental description, which mentions the *posture*, a more obvious detail than the beard or the trousers (which are not easily visible).

When we create the meta-algorithm, we will take into account the fact that it is better to mention a nearby person if they stand out. We will be able to find out if the person stands out or not by running the algorithm on the people that are next to or near the referent. If the algorithm is able to describe one of those nearby people using no more than two attributes (the *type* and one more), we will consider that person easily distinguishable, because this means that there are no more characters in the scene of that *type* or with that second attribute. If there are several people

nearby that stand out, we will mention the one that is closest to the referent.

The Nearby People Algorithm, as it is now, mentions the person that is closest to the referent. We will modify it so it mentions the person that stands out the most instead.

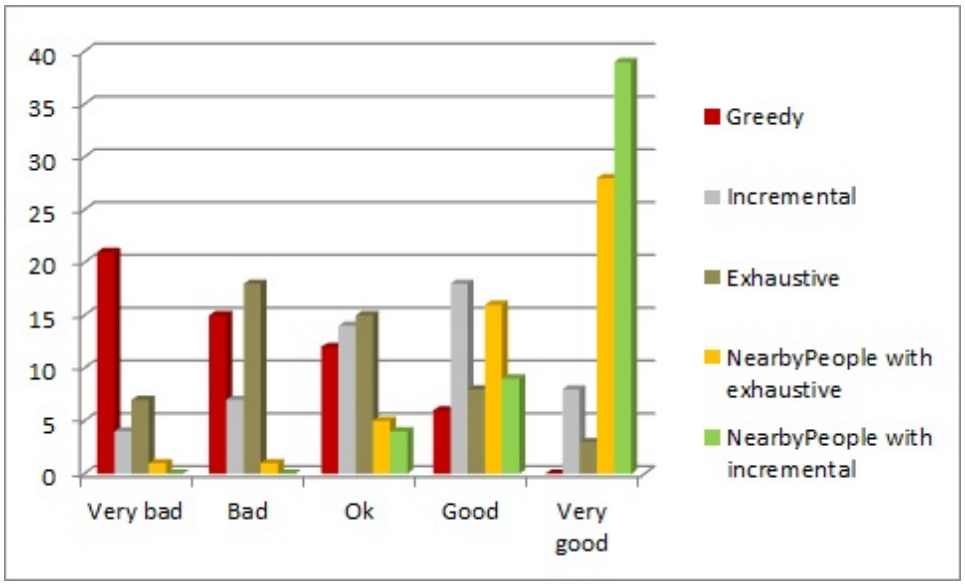

Figure 7.7: Second part of the second survey, results for question 1

7.4.2 Question 2:Rate these descriptions (Figure 7.8)

Figure 7.8: Second part of the second survey, question 2

The results obtained are the following:

- Greedy: *"The girl sitting down"* 74 % bad or very bad. 13 % good or very good
- Incremental: *"The girl in the white tank top who is sitting down. She is near."* 67 % bad or very bad. 26 % good or very good
- Exhaustive: *"The girl with medium length brown hair, with the white tank top and blue trousers."* 71 % bad or very bad. 12 % good
- Nearby Objects with Greedy: *"The girl sitting down near the window."* 36 % bad or very bad. 32 % good or very good
- Nearby Objects with Incremental: *" The girl in the white tank top who is sitting down. She is near. She is near the window."* 37 % bad or very bad. 38 % good or very good
- Nearby People with Greedy: *"The girl sitting down next to the boy in the dark blue sweater."* 60 % bad or very bad. 12 % good or very good
- Nearby People with Incremental: *"The girl in the white tank top who is sitting down. She is near. She is next to the boy in the dark blue sweater."* 34 % bad or very bad. 46 % good or very good

We can see that in this particular scene the posture alone is not useful, because there are too many people who are sitting down.

We cannot draw any definite conclusions from this question, because most of the users have confused the referent's skin as her clothes. When they read in the description that the girl's top is white, they were unsure about who we were referring to, because they thought that her top was pink. We think this confusion is mostly due to the size of the picture and the fact that it lost some amount of quality when we uploaded it. Also, if any of the users answered the survey from the small screen of their mobile phones, the confusion might have been even greater.

The results can be seen in figure 7.9.

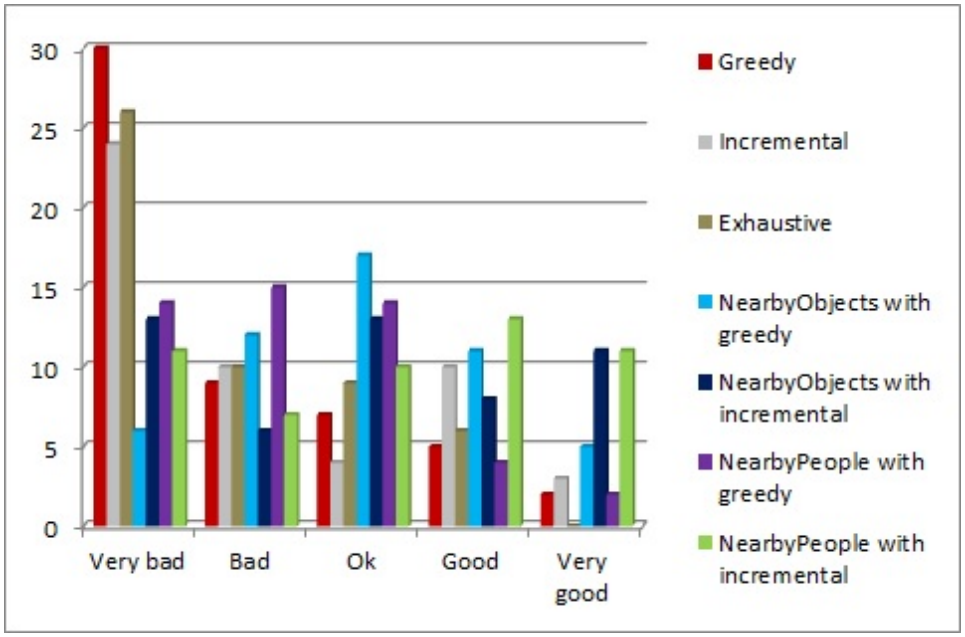

Figure 7.9: Second part of the second survey, results for question 2

7.4.3 Question 3:Rate these descriptions (Figure 7.10)

Figure 7.10: Second part of the second survey, question 3

The results obtained are the following:

- Greedy: *"The boy in the white shirt standing up, leaning on a table."* 16 % bad or very bad. 62 % good or very good
- Exhaustive: *"The boy in the white shirt, he has a beard and short black hair and he is wearing gray trousers. He is near."* 36 % bad or very bad. 32 % good or very good
- Nearby Objects with Greedy: *"The boy in the white shirt standing, leaning on a table. He is near the window."* 4 % bad. 78 % good or very good
- Nearby Objects with Incremental: *"The boy in the white shirt, he has a beard and short black hair and he is wearing gray trousers. He is near. He is near the window."* 14 % bad or very bad. 68 % good or very good

In this case the results show us (figure 7.11) that the Greedy description has had a large approval, both in its basic form and mixed with the Nearby

Objects Algorithm. This is due to the fact that both of them mention the *posture*, which is a particularly descriptive attribute. The Nearby Objects with Incremental algorithm has also had a good response, even though it does not mention the *posture* and it uses irrelevant attributes such as the beard, the length and colour of the hair and the colour of the trousers.

One last observation that we can make in this question is that we should modify the type of the clothes to only consider short sleeves (t-shirts) and long sleeves (sweaters). If we consider shirts and sweaters as different items of clothing, mentioning a person's shirt would rule out everybody with a sweater. Since in the scene the only difference that can be appreciated is between short and long sleeves, we will consider only three types of clothes: tank tops, t-shirts and sweaters.

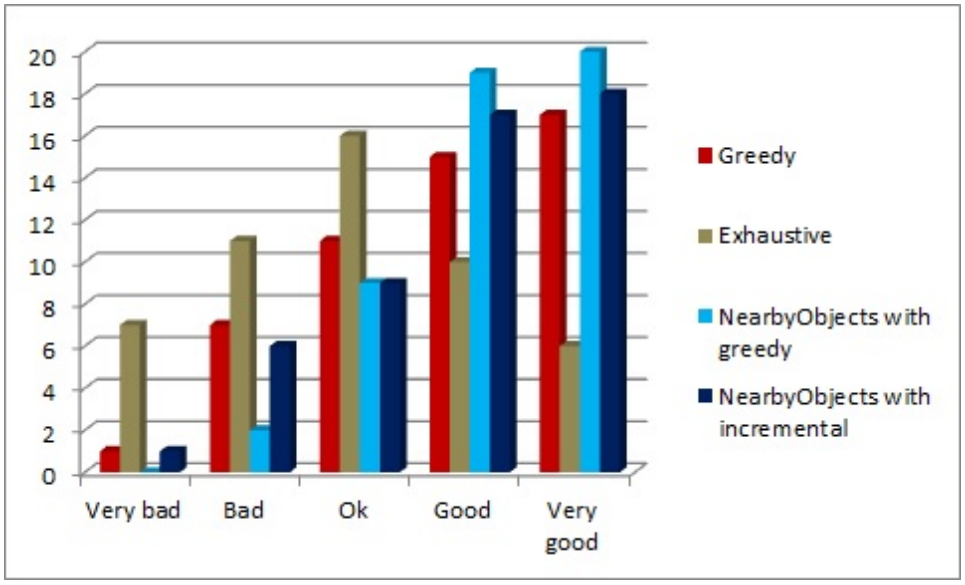

Figure 7.11: Second part of the second survey, results for question 3

7.4.4 Question 4:Rate these descriptions (Figure 7.12)

Figure 7.12: Second part of the second survey, question 4

The results obtained are the following:

- Greedy: *"The boy in the dark green t-shirt."* 56 % bad or very bad. 22 % good or very good
- Incremental: *"The boy in the dark green t-shirt who is sitting down, he has a beard and short black hair and he is wearing dark trousers. He is near."* 56 % bad or very bad. 18 % good or very good
- Nearby Objects with Greedy: *"The boy in the dark green t-shirt. He is near the last column."* 2 % bad. 86 % good or very good
- Nearby People with Greedy: *"The boy in the dark green shirt. He is next to the the black boy."* 28 % bad or very bad. 40 % good or very good

In this scene we chose a character who could easily be identified by himself, but who is also close to other characters and is close to one of the pillars, an important object in the room.

The results (figure 7.13) show us that in this case it is not very useful to describe only the referent. It has also not been very useful to describe

him in relation to a nearby person, because in this case *the black boy* is described, and since only his *type* is mentioned, he is not too easy to find. This reenforces what we mentioned in the first question of the second part of the test (section 7.4). Whether a character stands out or not depends on the number of attributes needed to describe them. We concluded that no more than two attributes should be used, but now we can see that the magic number is exactly two. If only the *type* attribute is used, it can take longer to find that person. If, however, the person can be distinguished by mentioning two attributes, we consider that this person stands out.

The relational description of Nearby Objects with Incremental, on the other hand, has worked very well. This is because the object is very easy to spot, and the Incremental algorithm provides the colour of the referent's top, which is important in order to tell him apart from the rest of the people in his group.

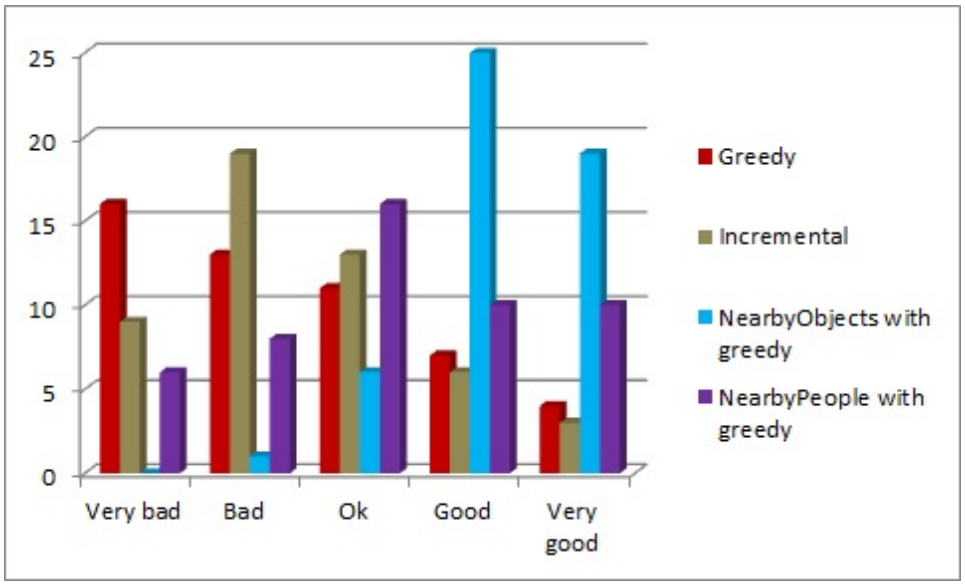

Figure 7.13: Second part of the second survey, results for question 4

7.5. Suggestions and observations

At the end of the survey we have included a small section for the users to make any suggestions or observations that they feel are important. These are the observations that they provided:

- As we have mentioned in question 2 of the second part of the survey (section 7.4), many users thought that the girl's description was wrong because they mistook her skin for clothes. A lot of people have mentioned it in this section, and others thought that it was a trick question and done on purpose.
- A lot of the users have mentioned that the beard and the colour of the bottom garment were not visible, so we have decided to remove these attributes. Most of the people mentioned that they could not see anybody with a beard. This is very curious, since in the first survey people saw beards on people that did not have one.
- Some colours can be easily confused because they are similar (dark grays, dark blues, dark greens, browns or blacks). This is a complicated issue, because people tend to see colours differently. Some people can distinguish more shades than others, and other people see blue where another one sees green. We have decided to name colours in the most generic way possible, and we have not considered different shades of the same colour. This way, the algorithms will not rule out a character based on the particular shade of their top, and the descriptions will be clearer for everybody.

7.6. Conclusions

We had expected the answers to this survey to be similar to the ones in the first survey, since the scenes are very similar. We expected the users to mention the referents' postures, the colour of their clothes and their beards as they did before. We thought that the Incremental algorithm would be well received because it simulates natural language quite well and it considers some attributes more important than others based on the results of the first survey. We expected the Greedy algorithm to be the best because it always shows the referent's most distinctive attributes, making the description slightly shorter but full of useful information. We expected the relational algorithms to be well received, as they allow users to identify the referent very fast and easily.

The results, however, prove that the users prefer the Incremental Algorithm rather than the Greedy Algorithm. But they do seem to prefer descriptions that mention nearby objects or nearby people. Whenever a person stands out, the users have immediately mentioned them. Large important areas of the room seem to provide good aids for the descriptions too.

While analysing the results from this survey, we have decided to give the attributes relative to the *hair* (*hair length* and *hair colour*) less priority in the algorithms. In these scenes, the length of the hair is always similar and the colour cannot be appreciated too well. The colour of the bottom garment in most cases cannot be seen very well, and even when it can be seen, the users do not use it very much in their descriptions, so we have decided to remove it from the algorithms. The same applies to the beard and we have also removed it. In the scenes we have created, the *beard* attribute may be misleading, because some algorithms are using it in cases where it is not clearly visible. This is due to the quality of the characters that we were able to download. Based on the results from the first survey we think that with better characters the *beard* attribute would be very informative.

Chapter 8

Second Iteration

After analysing the results obtained in the second survey, we are going to focus on improving the Nearby People and Greedy Algorithms. This will allow us to generate better referring expressions. Once all the algorithms are ready, we will create the meta-algorithm based on the conclusions we drew and the improvements we thought of in the previous survey.

In order to make it easier to manage the source code and to add scenes and other mechanisms to the application, we will reorganise the execution flow of the application and make some changes to the architecture. We will apply all the knowledge we have gathered on the resources provided by Unity.

8.1. Final modifications of the algorithms

In order to improve the performance of the Nearby People algorithm, we have decided to change one detail in its behaviour. Based on the results of the previous survey, we can see that it is not always the closest person that gets mentioned, but the one that stands out the most. As we have already explained, it is a better solution to check how many attributes are needed to describe the person with the Greedy algorithm. The one that needs the least attributes is the one that stands out the most, and more specifically the one that uses only two attributes will be the best. This is because using only one attribute (the *type*) is sometimes not enough information to find the referent quickly. For this person we have added an extra parameter to the Greedy and the Incremental algorithms, which returns the number of attributes they have used to generate the description. In the case of the Incremental Algorithm, it first runs the Greedy algorithm to calculate the number of attributes needed, and then generates the Incremental description.

Also, as we explained while analysing the results of the second survey, we have removed the *beard*, *bottom type* and *bottom colour* attributes. We have

also given less priority to the *hair length* and *hair colour* in every algorithm, since they have not proved very useful in our scenes.

8.2. The Meta-Algorithm

The Meta-Algorithm is based on the algorithms that we created in the previous iteration and the score that each of them got in the users' ratings during the previous survey. The purpose of this algorithm is to choose which of the existing algorithms it should use depending on the situation.

First the Meta-Algorithm tries to create a Nearby People with Incremental description. This algorithm has been modified to first run the Greedy algorithm, as we have mentioned above. If it determines that there is a nearby person that is very easily identifiable (they can be described by using only two attributes) it will return that description. We have chosen this algorithm because the users have preferred the Incremental Algorithm over the Greedy Algorithm, and we have concluded that it is preferable to mention a nearby person if there is one that stands out enough.

If there is no other character nearby that is sufficiently distinguishable, the Meta-Algorithm goes on to find out if the referent stands out in the scene. By running the Greedy Algorithm, we find out if they can be referred to by using exactly two attributes, and in that case the referent is easily distinguishable. We can now generate the Incremental description for the referent, again, because the Incremental Algorithm is better than the Greedy Algorithm and provides more information.

If the referent does not stand out, the Meta-Algorithm will now try to use the Nearby Objects with Greedy Algorithm. We use the Greedy Algorithm here to describe the referent, because we consider that the object is descriptive enough, and we do not need any extra information about the referent.

If there are no objects near the referent, the Meta-Algorithm will finish by generating the description with the Greedy Algorithm.

8.3. Final changes in the architecture

Up until this point, the application consisted only of one scene that generated and deleted all the characters for each question. This makes modifying the program and adding more scenes quite difficult, and it means the camera has to be manually set in different coordinates after every ques-

tion. It also makes it harder to add different objects to the scene or modify the canteen (move the chairs or tables around). We also had to add a very large black cube to cover the scene while the user was reading the description.

We will now change the architecture so that we have different scenes for every question, and an extra black scene with the *Start* button and the description of the referent before we load each question. This makes it a lot easier to modify the application if more scenes have to be added to it.

On the other hand, it also has a few disadvantages. The application is less efficient this way, because the description must be generated in the black scene, before the question is loaded, and it takes a few moments for it to finish. It also makes running the application on a browser slower, because there are now more scenes to be loaded.

8.4. PHP, running the application on a browser

After speaking to some of the users, we realised that they are quite lazy and that they prefer not to have to write too much or go through a lot of steps in order to answer our surveys. For this reason, we decided that we should enable this application for its use in a browser, because if we were to send potential users an executable file, most people would not bother opening it. The server we will use is from our own university, `tot.fdi.ucm.es`. In order to run this project from a browser, the user must have Unity Web Player installed. Even so, the program can still be compiled into an executable file and can be run on Windows, Linux or Mac.

An easy way to communicate from Unity's web solution for browsers to the server is by using a kind of scripts called *PHP*. This file has to be included in the same domain as the application due to the security restrictions. The script will be run once a petition is made to the address `http://tot.fdi.ucm.es/descripciones/script.php`. This can be done writing this address where the file is located, followed by the symbol "?" to add any parameters. The parameters should be separated by the symbol "&" and will have the following structure:

name=value

This will allow us to gather the data from the program's execution and save it inside a *txt* file in the server, so we can later analyse the results in the simplest way possible. In our case, we organised the structure of the file so we could easily analyse it using *Excel*, because it allows us to program simple methods to make the statistical calculations that we need.

8.5. Conclusions

Even though the disadvantages that have come from the changes in the architecture are not an important problem, it would be possible to optimise the application further. This, however, would take a lot of effort and time that we do not have. The advantages of those changes have made the programming a lot easier for this project and for any future modifications that anybody may wish to make.

Chapter 9

Third survey

9.1. Purpose of this survey

The purpose of this last survey is to test the effectiveness of the Meta-Algorithm.

The survey has two parts. In the first part we collect some basic data from the user, and in the second one we test the application. In order to do this, the user must have Unity Web Player installed, and they must run the application from their browser. Some minor issues have been experienced on a few occasions with certain browsers. This is due to their incompatibility with Unity and this type of application. In most cases, however, the application works well with Google Chrome, Mozilla Firefox and Internet Explorer.

We have a total of three scenes, so we have shown each of them three times to create nine different questions. We always show the three scenes in the same order, so it is less likely they will remember the scene when they see it again. As for the descriptions, for each scene we have chosen one where the Meta-Algorithm generates a basic referring expression, one where it generates a referring expression with a nearby object, and one that uses a nearby person.

9.2. Links

The application can be found in a server in our university. The link is:
`http://tot.fdi.ucm.es/descripciones`

The raw data collected after each user's response can be found in a file in the same domain as the application. The link is:
`http://tot.fdi.ucm.es/descripciones/data.txt`

9.3. Part one: demographic data

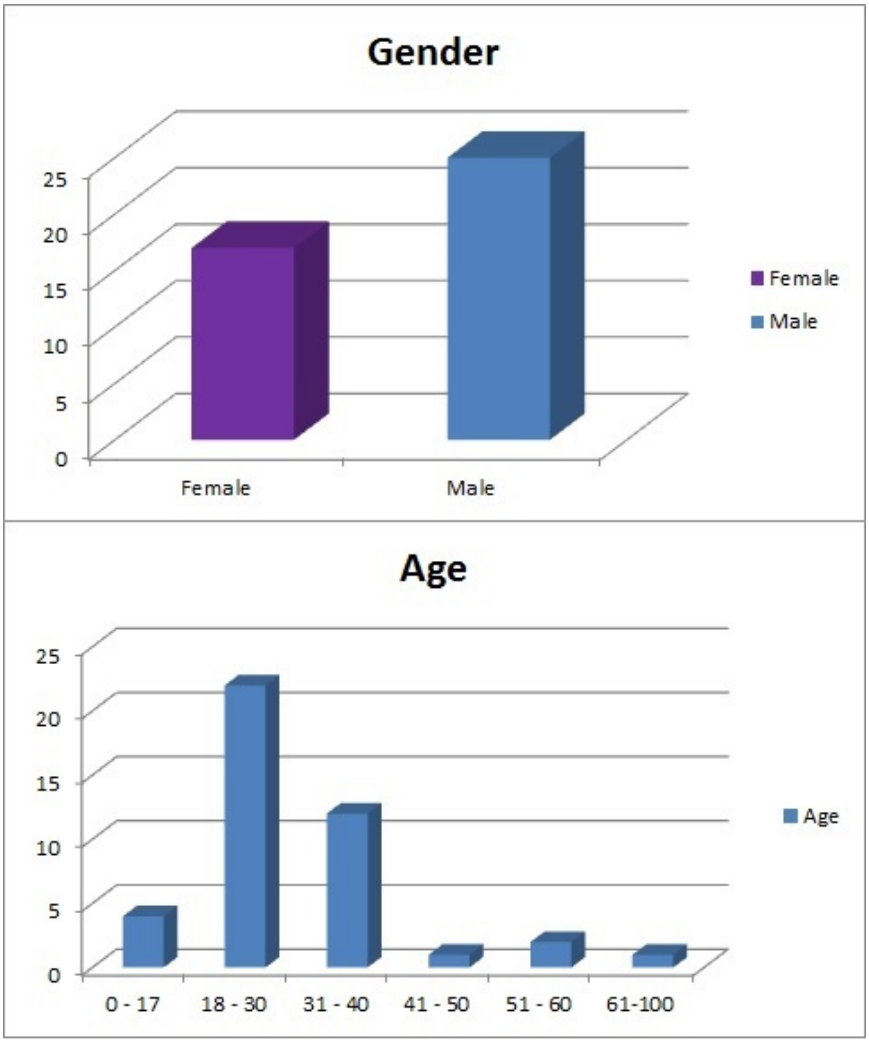

Figure 9.1: Gender and age distribution for the third survey

In this survey, we have gathered the answers of seventeen women (40 %) and twenty-five men (60 %). A total of forty-two people have completed the test. The ages vary, although most users (81 %) are in between the ages of eighteen and forty. We have observed that the younger the person is, the more likely it is that they answer the questions incorrectly. This might be due to a poor understanding of the descriptions provided, or to the fact that they might be more impatient and in more of a hurry to finish.

9.4. Part two: find the person

All the scenes used in this part of the survey can be seen at the end of section 9.4.

9.4.1 Scene 1 (Figure 9.11)

The description provided in this scene is: "The boy in the black sweater."

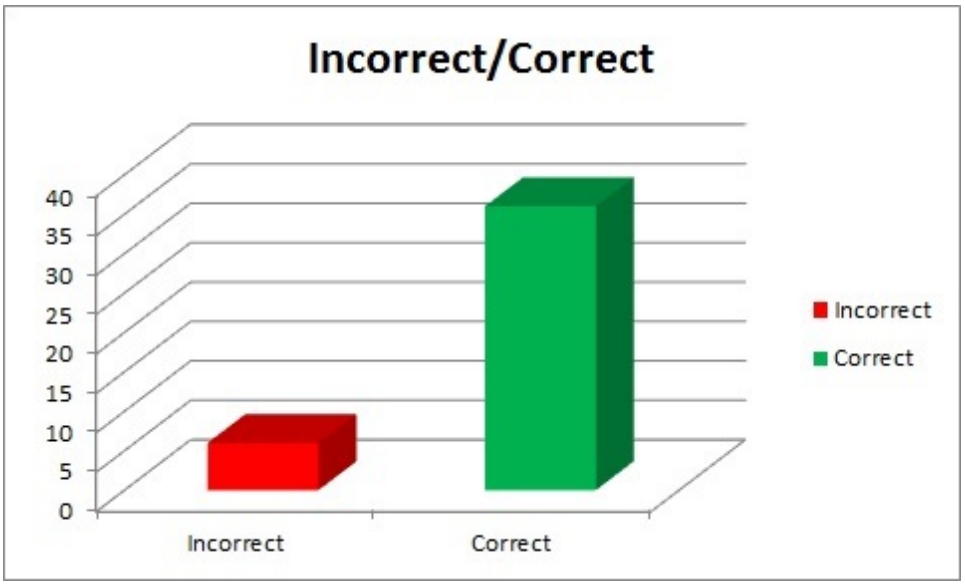

Figure 9.2: Second part of the third survey, results for question 1

All the people who have answered incorrectly (14 %), except one person who has chosen a completely different character, have chosen another boy for whom the same referring expression could apply, but who is at the far end of the room. It has taken almost everybody in between twenty-two and forty-two seconds to answer. Results can be seen in Figure 9.2.

Since we have based our algorithms on the previous surveys, we noticed that people almost always focus on the characters that are closest to them, so we did not need to mention the distance at which the referent was in order to achieve a very high hit rate. We consider that the mistakes in this case are justified, since the description could apply to both characters. In order to avoid this confusion, the Greedy Algorithm (which is the one used in this case) could be modified to include the distance of the referent.

9.4.2 Scene 2 (Figure 9.12)

The description provided in this scene is: "The boy in the green sweater near the window."

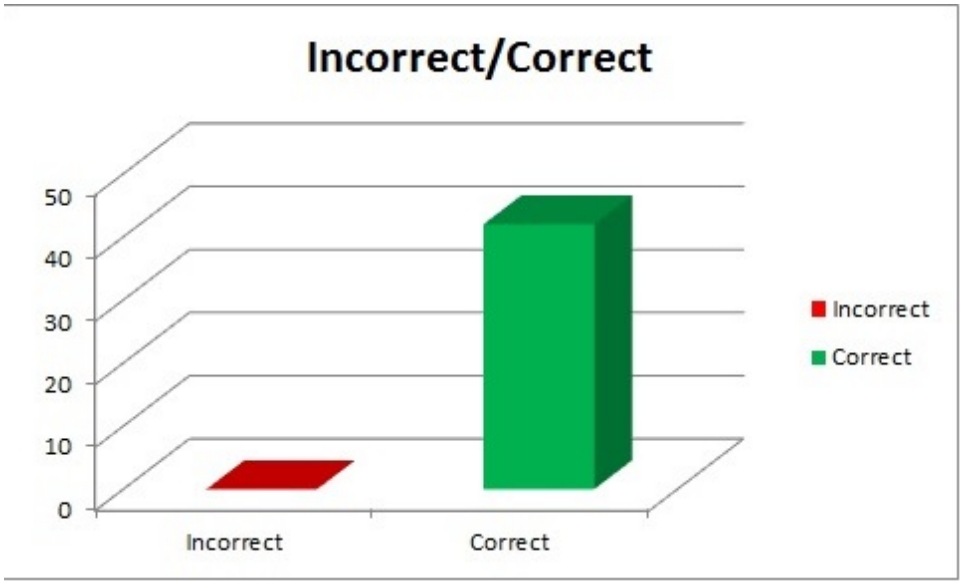

Figure 9.3: Second part of the third survey, results for question 2

In this scene everybody answered correctly. The referent is the only person dressed in green, so he is easily recognisable. The average response time for this question is lower than expected. This could be because the person is close to the window, which makes the search easier, because it gives the user an area to focus on. Results can be seen in Figure 9.3.

9.4.3 Scene 3 (Figure 9.13)

The description provided in this scene is: "The boy in the gray sweater. He is near. Next to the the boy in the white t-shirt with circles."

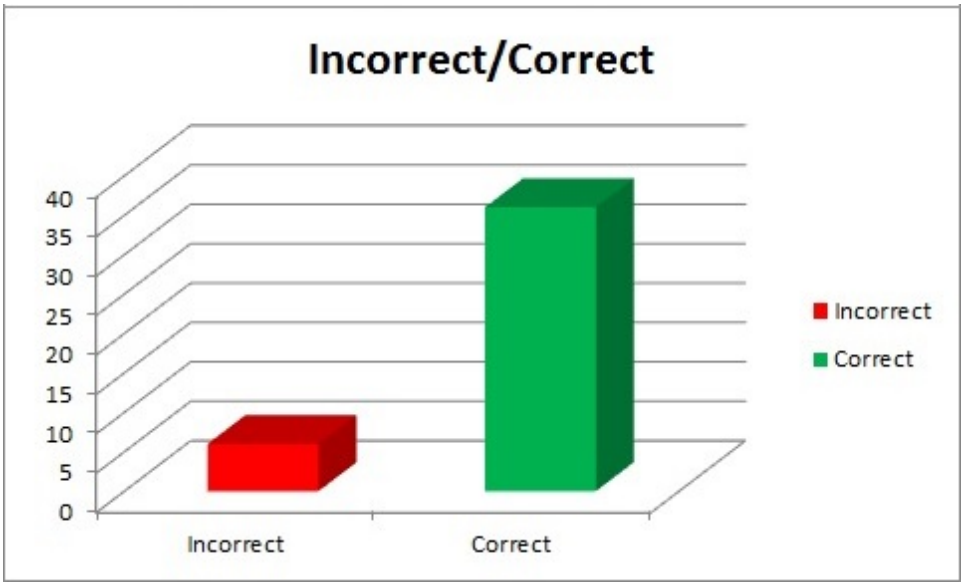

Figure 9.4: Second part of the third survey, results for question 3

This scene has a miss rate of 14 %. Two users have chosen a boy dressed in black who is close to the observer. They may have confused the black sweater as a dark gray. These two people have taken quite a long time to answer (31 seconds). Three users have chosen a boy who is close to the observer and is dressed in dark blue. This character is sitting next to a boy in a white sweater with a black pattern (not circles). This may be because the last thing that the users read was the description for the nearby person, and they might remember this better. They might have chosen this person because he is a lot more visible than the referent, who is partly hidden behind another person. Results can be seen in Figure 9.4.

9.4.4 Scene 4 (Figure 9.11)

The description provided in this scene is: "The boy in the white t-shirt standing near the bar."

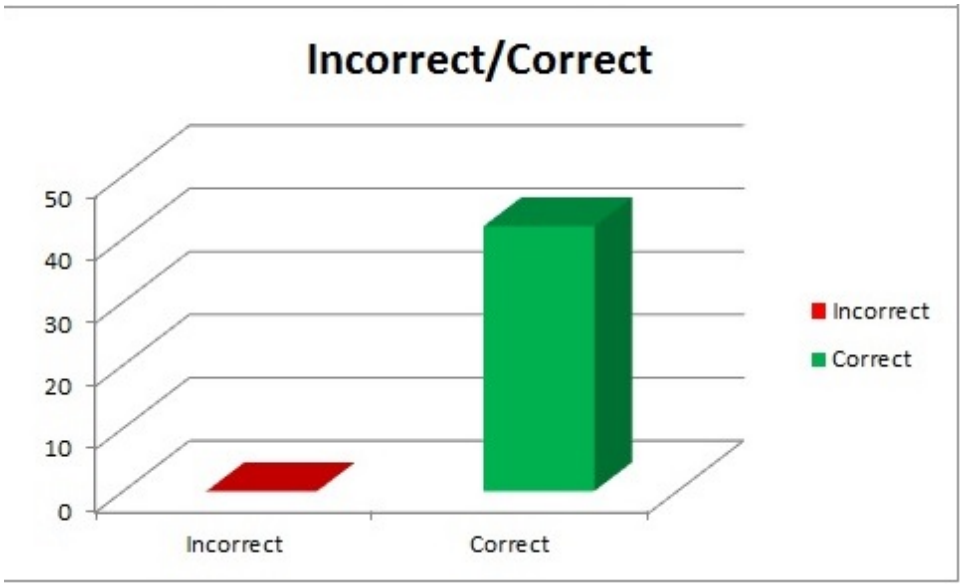

Figure 9.5: Second part of the third survey, results for question 4

In this scene we have also achieved a 100 % hit rate. Just as in the second question, where everybody answered correctly, the algorithm that has been used has been the one that mentions Nearby Objects. We have obtained a good average time as well, slightly lower than in question 2. Results can be seen in Figure 9.5.

9.4.5 Scene 5 (Figure 9.12)

The description provided in this scene is: "The girl in the yellow and brown t-shirt who is standing up with black hair. She is near. Next to the the boy standing pointing at something."

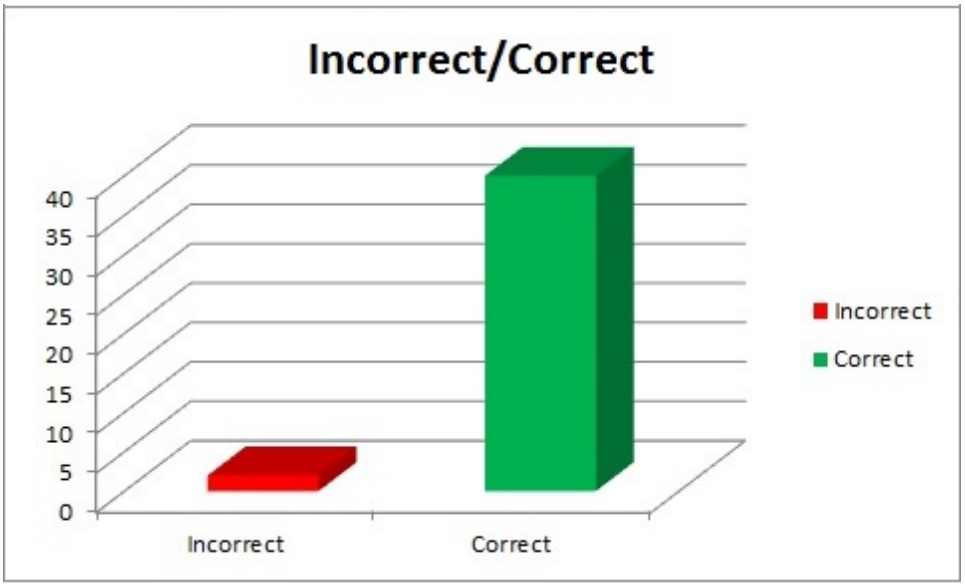

Figure 9.6: Second part of the third survey, results for question 5

The average response time for this question is higher, and this might be because the description provided is considerably longer.

There are two incorrect answers for this question. The first one has chosen a boy dressed in yellow who was not very visible. This might be because the user mistook him for a girl (he is facing away from the observer) or because they simply clicked in the wrong area (both characters are quite close together). The other user has chosen a character that does not fit into the description at all, the only thing it has in common with the referent is the gender. The character that this person has chosen is closer to the camera. Results can be seen in Figure 9.6.

9.4.6 Scene 6 (Figure 9.13)

The description provided in this scene is: "The boy in the black and white sweater who is sitting down with short brown hair. He is near."

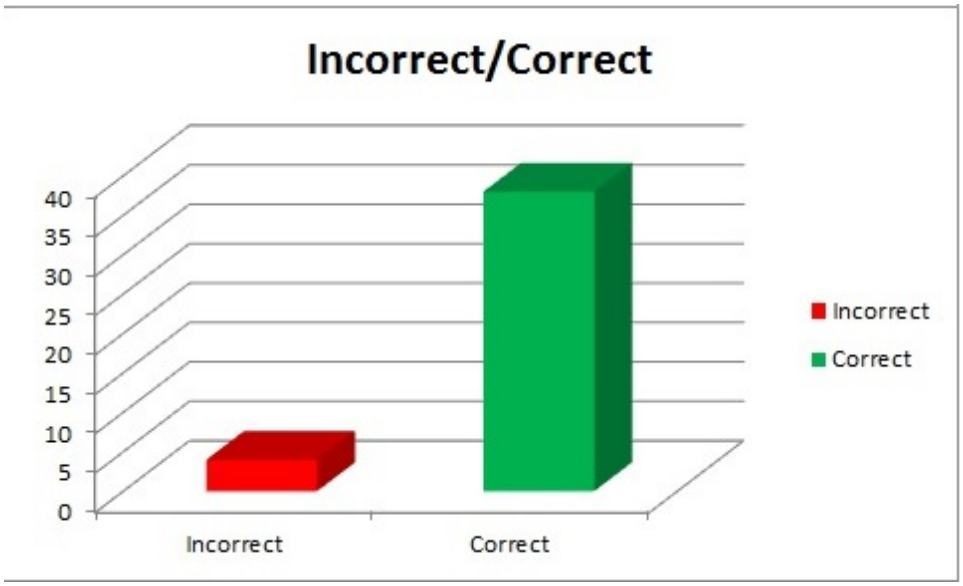

Figure 9.7: Second part of the third survey, results for question 6

There are four people who have answered this question incorrectly. Two of them have chosen another boy who also has a black and white sweater, who is in the center of the scene, partly covered by another character, but he is standing up, not sitting down. These answers have a response time (4.3 and 6.8 seconds) considerably lower than the average (8.5 seconds). This might indicate that the users answered quickly without reading the description thoroughly. The other incorrect answers also have a low response time (5.1 and 4 seconds) and the chosen characters are not similar to the referent. However, they are quite close to him and to the other distractor that the people who were wrong selected. Results can be seen in Figure 9.7.

9.4.7 Scene 7 (Figure 9.11)

The description provided in this scene is: "The boy in the red sweater with short black hair. He is far. Next to the the boy in the black and grey striped sweater."

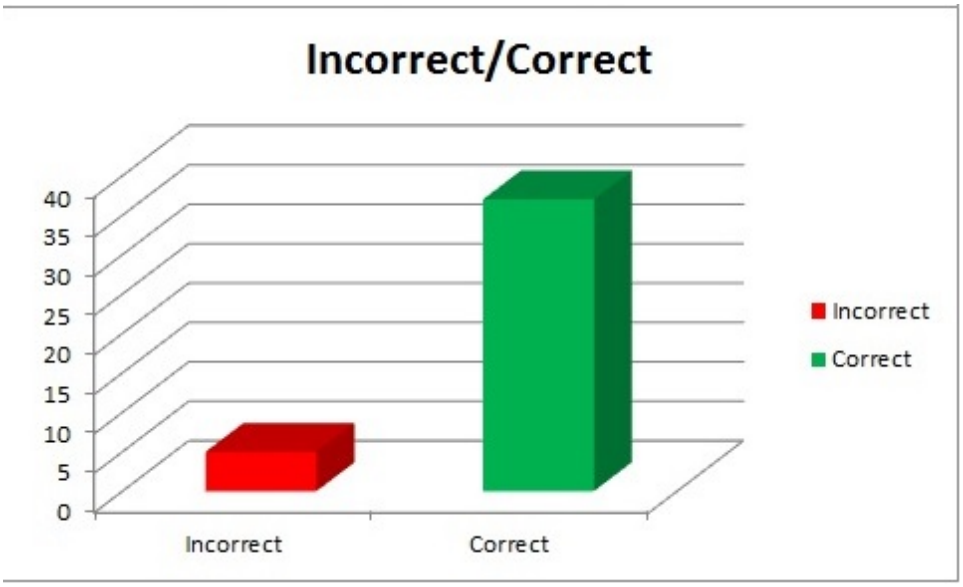

Figure 9.8: Second part of the third survey, results for question 7

In this scene, less than 12 % (five people out of forty-two) were wrong. Most of them have chosen another boy wearing a red sweater who is a lot closer to the observer and more visible. Their response times have been quite lower (4.7, 1.6, 5.7 and 4.7 seconds) than the average (7.1 seconds). We think they may not have taken into account the reference to the nearby person and they have answered quickly, without reading the whole description. The other user who made a mistake also chose a boy in a red sweater, but he is not very visible, and we do not see the reason for their confusion. Results can be seen in Figure 9.8.

9.4.8 Scene 8 (Figure 9.12)

The description provided in this scene is: "The girl in the white t-shirt who is standing up with blonde hair. She is near."

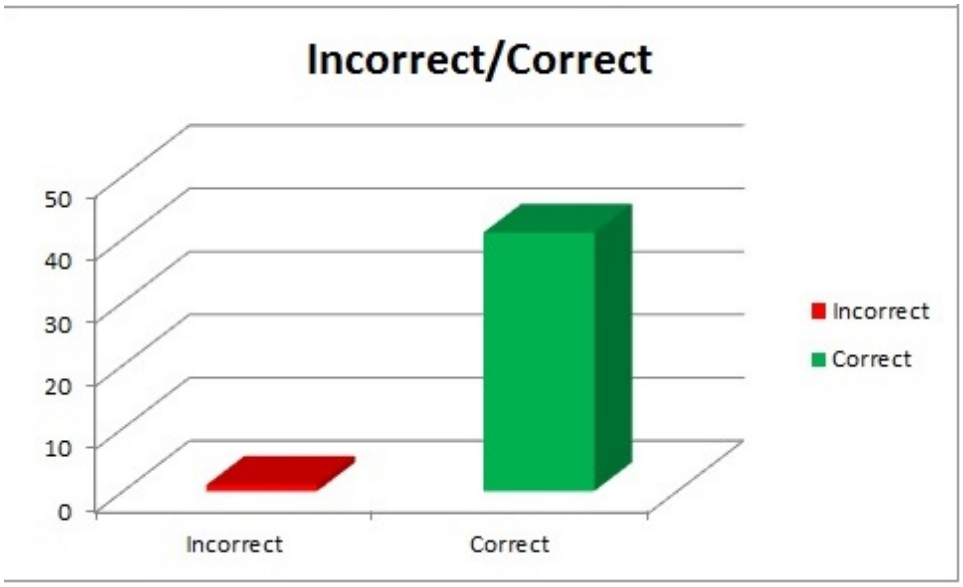

Figure 9.9: Second part of the third survey, results for question 8

Only one user has answered incorrectly. They clicked on a boy in a blue sweater who is sitting next to a girl. The description does not match the referent's and they do not look similar, so we do not understand this user's confusion. Results can be seen in Figure 9.9.

9.4.9 Scene 9 (Figure 9.13)

The description provided in this scene is: "The boy in the white sweater standing, leaning on a table near the window."

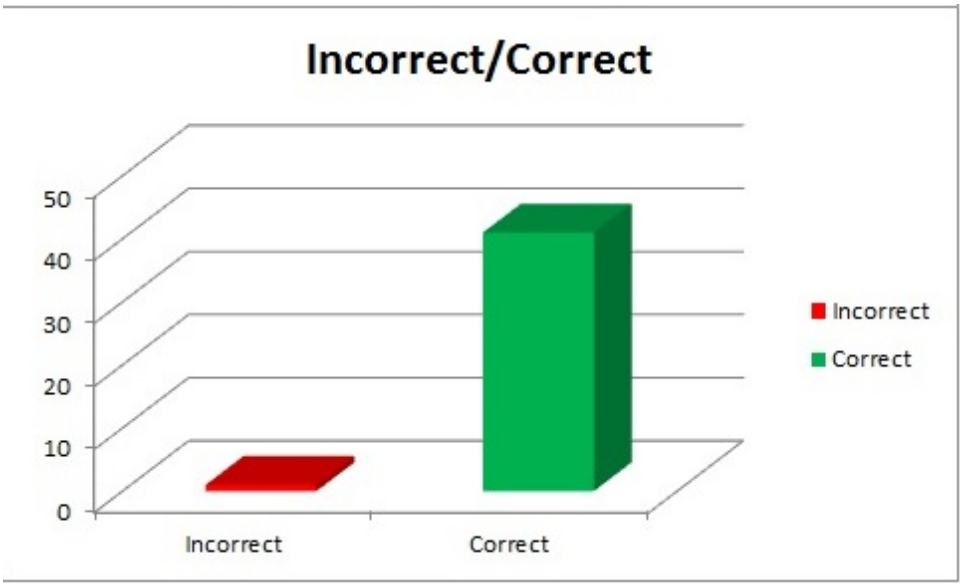

Figure 9.10: Second part of the third survey, results for question 9

Just like in the previous scene, only one person has made a mistake. They have chosen a character who is also dressed in a white sweater and who is close to the window, although not as much as the referent. He is sitting down and leaning on the wall, so the description does not match the one provided. The response time is, again, slightly lower than the average. Results can be seen in Figure 9.10.

Figure 9.11: Second part of the third survey, scene 1 (questions 1, 4 and 7)

Figure 9.12: Second part of the third survey, scene 2 (questions 2, 5 and 8)

Figure 9.13: Second part of the third survey, scene 3 (questions 3, 6 and 9)

9.5. Conclusions

By observing some of the users and talking to them after the test, we can see that many of them were not fully focused on the survey, because they did not know that they were being timed. Some of them do several other activities at the same time as they are answering the questions. On some occasions, the longer descriptions are harder to remember and the user needs to read them again. This usually happens while the scene is already being shown and the timer is running.

The error times are very high (higher than twenty-two seconds) or very low (less than five seconds), with a few exceptions. As the users move on in the survey, the response times get lower and lower, from between 66 and 42 seconds to around 12 seconds. The shortest response time remains close to 1 second.

The total hit rate is 93.4 % (353 correct answers out of 378), so there is 6.6 % error rate (25 incorrect answers out of 378). With these results, it seems that our Meta-Algorithm cannot be improved too much further with our algorithms.

It is worth noting that when we have used the Nearby Objects Algorithm, there has only been one mistake (in the third scene that uses this algorithm). The error rate of this algorithm is less than 0.8 %, so we can conclude that if there is an easily recognisable object near the referent, the users will find them correctly. A reasonable modification in

the Meta-Algorithm would be to give the Nearby Objects Algorithm more priority than the Nearby People Algorithm.

A good alternative that would improve the behaviour of the application and would simulate a real conversation, would be to create a mechanism that complements the referring expression and provides more information about the referent if the user guesses incorrectly. This way they could have more than one chance to answer.

Chapter 10

Individual Work

This final project has two authors. Some parts have been developed by both of them together, and others have been done individually. Here we will review the work that each of the members has done.

10.1. Adrián Rabadán Jurado

First of all, since we knew that the project would be developed with Unity, we both familiarised ourselves with its environment by completing tutorials to understand what Unity is capable of doing, how to incorporate code to the application, and the script languages that can be used with it.

We had to take the appropriate pictures to research and create the first survey (pictures of the canteen in our university while it was full). After waiting for a few days to gather enough data from the users, we both analysed the results from the survey. Adrián analysed the second half of the results and created some graphs to visualise the data.

Adrián also created and designed the first structure of the XML file that would hold the characters' information. Later, based upon this design, we created a DTD file that checks whether the structure of the XML file is correct.

Once the characters and their attributes were finished, he focused on creating the Nearby People and Nearby Objects algorithms. This version of the algorithms mentioned the person or object closest to the referent.

In order to start testing the application in an environment similar to the final one, he created a temporary room (a rectangular shape with lights). This would allow us to see the character's sizes and the distance at which

they should be.

Some of the users complained after the first survey that it took too long to complete. We realised that we would have to set up the application so that it could be run from a browser. Adrián started preparing the project for this, including all the information that needs to be loaded in a *Resources* folder, and configuring the project correctly.

Before sending out the second survey we added a large amount of characters to the scenes. With all of these new characters, we detected some mistakes we had made in some of the algorithms (the Incremental and the Greedy) and Adrián fixed them.

Next, with all of the algorithms ready, we prepared the second survey, taking screenshots of the generated scenes and adding the necessary details. This survey consists of two parts. In the first part we ask the users to give us a description for three specific characters. In the second part we ask them to rate some descriptions generated by our own algorithms. This last part was the most important, because it helped us realise that the descriptions that the users generate are often very different from the descriptions that they consider to be good. Adrián created new graphs to show the results for this survey. In order to make the process of choosing the descriptions we would use in the survey easier, he designed a method that creates a text file with all the different descriptions generated for each character in each scene.

We both analysed the results of the second survey and discussed how the Meta-Algorithm should work. Before implementing it, Adrián corrected a few mistakes in the previous algorithms and modified their behaviour according to our conclusions from the surveys. He then went on to implement the Meta-Algorithm based on the results obtained from the second survey.

While preparing the last survey, he learnt the mechanisms that Unity has in order to send data and petitions to a server. The simplest way that could work for our purpose consisted in learning a scripting language called PHP that allows files to be created and modified. He programmed one that would store the data in the server when the survey had been completed by each user, in a way that would allow it to be later analysed easily using Excel.

When we prepared the last survey, which makes use of the final application, we thought it would be reasonable to reorganise the structure of the project in Unity so that it would be easier to add more scenes to the program or modify existing ones. He included a mechanism that randomly

selects a referent from a group of possible candidates given for each scene, and the possibility to add manual translations of the resulting descriptions into Spanish. This way it would be easier to gather results from the survey.

Lastly, he analysed the results of the third survey and generated the appropriate graphs with Excel.

He also wrote some chapters of this document in Spanish.

10.2. Teresa Rodríguez Ferreira

First of all, since we knew that the project would be developed with Unity, we both familiarised ourselves with its environment by completing tutorials to understand what Unity is capable of doing, how to incorporate code to the application, and the script languages that can be used with it.

Teresa researched the existing algorithms in the field of the Referring Expression Generation, reading a number of articles and surveys. We then discussed how we would make our own versions of the algorithms. Teresa also included all the research gathered in chapter 3.

We had to take the appropriate pictures to research and create the first survey (pictures of the canteen in our university while it was full). Teresa put together the first survey, choosing the people we would select for it and putting together the photographs. After waiting for a few days to gather enough data from the users, we both analysed the results from the survey. Teresa analysed the first half of the results.

Teresa found some free characters in the Unity Asset Store and started investigating how we could modify and personalise them. She created some scripts for testing that allowed us to modify the character's postures and attributes.

Once the characters and their attributes were finished, she focused on creating the Exhaustive algorithm. This version of the algorithm is the same one that we have now.

After all the basic algorithms were done, Teresa created the Incremental Algorithm and the Greedy Algorithm. These were based on existing algorithms and modified to fit in with our application and to respect the conclusions we drew from the previous survey.

Teresa created all three scenes in Unity, modifying the characters

and the furniture to imitate the photographs taken at the beginning of the project. She used Photoshop to create different types of clothes for the characters. In order to easily set all the characters' postures from a script, she created a method that read the rotation and coordinates of all the characters and their articulations, and this method would generate the code that would allow us to modify their postures from a script.

Before sending out the second survey we added a large amount of characters to the scenes. With all of these new characters, we detected some mistakes we had made in some of the algorithms (the Incremental and the Greedy). Teresa fixed some of these mistakes.

Next, with all of the algorithms ready, we prepared the second survey, taking screenshots of the generated scenes and adding the necessary details. This survey consists of two parts. In the first part we ask the users to give us a description for three specific characters. In the second part we ask them to rate some descriptions generated by our own algorithms. This last part was the most important, because it helped us realise that the descriptions that the users generate are often very different from the descriptions that they consider to be good. Teresa chose the characters we would use in the survey that would be most representative and would provide the most information.

We both analysed the results of the second survey and discussed how the Meta-Algorithm should work. While Adrián implemented the Meta-Algorithm, Teresa continued writing this document, at this point focusing on finishing the chapter dedicated to the First Iteration.

When we prepared the last survey, which makes use of the final application, we thought it would be reasonable to reorganise the structure of the project in Unity so that it would be easier to add more scenes to the program or modify existing ones. Teresa helped to make sure that all the scenes in the last version of the project were correct.

Lastly, she finished writing this document, translating the parts that Adrián had written in Spanish and writing the rest of the document herself.

Chapter 11

Conclusions and Future Work

11.1. Conclusions

The representation in 3D of a real situation is not easy. In general, the fewer resources available for the creation of that scene, the harder it will be to identify the people and their features. Depending on the type of screen being used to visualise the scene, the size of the screen, and its quality, sometimes the people or objects in it cannot be properly recognised. We also have to consider that properties such as colours can easily be confused, or interpreted in different ways by people. For each person a different shade can represent a different colour. Another problem to be considered is that daltonic people see colours in a completely different way.

When trying to find a person or an object in a scene, especially if the scene is crowded, the visibility of that person or object plays an extremely important role. The less we can see, the worse we will be able to recognise a person and the harder it will be to find them.

A person's reading comprehension can also interfere with the objective of referring expressions, which is none other that to recognise and identify, out of many distractors, a particular referent. When we analysed the results of the last survey, we noticed that the users' reading comprehension improved with age.

The shorter the referring expression, the easier it will be for the user to recognise the referent, because when a description is long, we are also testing the user's memory. Referring expressions should be kept short, without compromising their quality. Descriptions that are too short are less informative and it will take the user longer to find the referent, but descriptions which are too long will confuse them. When a referring expression contains extra information, even if this information is not indispensable in

order to recognise the referent, it becomes easier to understand and sounds more natural. We demonstrated that this is true with our second survey, in which people always preferred the Incremental Algorithm over the Greedy Algorithm.

We have concluded that by including in the descriptions references to other objects or people that the referent is related to, we can considerably improve the quality of the referring expressions, but their length needs to be balanced. As we mentioned before, if the description is too long, it becomes confusing and hard to remember and the user will need to re-read it.

Considering all these conclusions, we have been able to create an application that is able to generate different referring expressions depending on the situation, with a very high hit rate (93.4 %).There is still room for improvement, of course, but we have been able to complete all our objectives for this project:

- With our research we have expanded the knowledge in the field of the Generation of Referring Expressions
- We have found patterns in the users' descriptions and we have discovered which details they notice the most about people, both real people and characters created with Unity
- We have created three basic algorithms that generate good referring expressions: the Exhaustive Algorithm, the Nearby People Algorithm and the Nearby Objects Algorithm
- We have created two more complex algorithms, the Incremental Algorithm and the Greedy Algorithm, and we have combined them with each of the basic algorithms
- The descriptions generated imitate natural language and are easily understandable
- Our scenes recreate the university canteen, and they are full of different characters that fit into everyday situations
- We have compared the algorithms to find out which work best in each situation
- We have created a Meta-Algorithm that is able to choose and combine the algorithms, to create the most suitable referring expression for each situation

11.2. Future Work

The field of the Generation of Referring Expressions has a lot to offer, and this project provides new insight and conclusions on the matter, but since we have very limited time to complete it, we will consider the application finished at this point. In case anybody wishes to continue this project, we will discuss several ideas that might be interesting to work on.

We would like to find out how much having higher quality characters would influence the users' responses. With better characters, details such as beards, different shades of colours or patterns on the characters' clothes could be better appreciated.

We would like to add objects to the scene, such as laptops, food, backpacks, or special items for the characters, for example a hat or a tie. The latter could be included in the XML file as *special* attributes.

It would be interesting to have more varied characters. We could have characters of different height, because it could be descriptive to mention if someone is tall or short compared to the rest of the characters in the scene. The same could be applied to their weight.

In the first survey we observed that one particular character was very easily distinguishable by her *type* (*"waitress"*). In our application we only used different *types* to refer to people of a different race (*"black boy"*) and when a referring expression used only the *type*, it was not very useful for the users. It would be interesting to see if this would change if we had more obvious *types*, such as *"waitress"* or *"cook"*.

We would like to be able to consider different tones of the same colour. It would be possible to divide the *colour* attribute into two: the colour and the shade (dark, light or simply neutral). This would allow the algorithms to rule out a character based on the colour of their clothes but not on the shade.

As to the user's interaction with the application, it would be interesting to implement a feature that simulates a conversation, as we suggested in the conclusions from the last survey. This way we could start by using the most descriptive attributes, and keep adding more information to the description until the user guessed the person correctly. This could mean implementing new algorithms and considering attributes in a different order. An example of the result that could be obtained is:

- *Do you see the boy waiting in line for his food?*

- *Which of them?*
- *The one dressed in blue*
- *Dark blue?*
- *No, light blue*

We would like to find out whether mixing nearby people and nearby objects in the same description would improve the results.

Another interesting thing to explore would be groups of people. Instead of simply considering individual people, we could mention that a person is sitting with a certain group of people.

Capítulo 12

Conclusiones y Trabajo Futuro

12.1. Conclusiones

La representación en 3D de una situación real no es sencilla. Por lo general, cuantos menos recursos se puedan destinar a la creación de dicha escena, peor se podrán reconocer los objetos y personas que haya en ella. Dependiendo del tipo de pantalla en la que se visualice, su tamaño y su calidad, se podrán distinguir o no los objetos y personajes de la escena. Hay que tener también en cuenta que características como los colores pueden ser fácilmente confundidas o pueden ser interpretadas de distinta forma por diferentes personas. Para cada persona un tono puede acercarse más a un color o a otro. A esta dificultad debemos añadir que los algoritmos implementados no tienen en cuenta que el usuario pueda tener problemas de daltonismo.

A la hora de localizar cualquier cosa en una escena, especialmente si la escena está muy poblada, la visibilidad de aquello a lo que nos refiramos toma un papel imprescindible. Cuanto menos se vea, peor se distinguen los rasgos de una persona y más dificil es encontrarla.

La comprensión lectora de una persona puede influir en el objetivo de las expresiones de referencia, que no es otro que reconocer y distinguir entre muchos distractores uno objetivo en concreto. Al analizar los resultados de la tercera encuesta, notamos que la comprensión lectora de los usuarios mejora con la edad.

Cuanto más corta sea la expresión de referencia más fácil le resultará al usuario reconocer al referente, porque cuando una descripción es demasiado larga, pondremos a prueba la memoria del usuario. Las expresiones de referencia deben mantenerse cortas, si por su longitud no se ve comprometida su calidad. Las descripciones que son demasiado cortas son menos informativas, y pueden hacer más complicada la tarea

de reconocer al referente, pero descripciones muy largas confundirán a los usuarios. Por otro lado, funciona mejor proporcionar una descripción algo más larga y que incluya algo de información no vital, antes que únicamente las características mínimas indispensables para distinguir al referente. Esto facilita la comprensión de la frase e imita mejor el lenguaje natural. Demostramos que esto es así en la segunda encuesta, en la que los usuarios preferían el Algoritmo Incremental antes que el Algoritmo Voraz.

Hemos comprobado que al incluir en las descripciones referencias a otros objetos o personas con los que mantiene una relación, la calidad de las expresiones de referencia aumenta considerablemente, aunque su longitud debe mantenerse equilibrada. Como ya hemos explicado, si la descripción es demasiado larga confundirá al usuario, tendrá problemas para recordara y seguramente tendrá que releerla.

Teniendo en cuenta estas conclusiones, hemos logrado crear una aplicación que es capaz de generar distintas expresiones de referencia dependiendo de la situación, con una tasa de aciertos muy alta (93.4 %). La aplicación se podría mejorar, por supuesto, pero hemos sido capaces de cumplir todos nuestros objetivos para este proyecto:

- Con nuestra investigación hemos contribuído al conocimiento existente en el campo de la Generación de Expresiones de Referencia
- Hemos encontrado características comunes en las descripciones de los usuarios, y hemos descubierto los detalles en los que más se fijan de las personas, tanto de personas reales como de personajes creados con Unity
- Hemos creado tres algoritmos básicos que generan buenas expresiones de referencia: el Algoritmo Exhaustivo, el Algoritmo de Personas Cercanas, y el Algoritmo de Objetos Cercanos.
- Hemos creado dos algoritmo más complejos, el Algoritmo Incremental y el Algoritmo Voraz, y los hemos combinado con cada uno de los algoritmos básicos.
- Las descripciones generadas imitan el lenguaje natural y son fáciles de comprender
- Nuestras escenas recrean la cafetería de nuestra facultad, y están llenas de personajes distintos que encajan con situaciones cotidianas
- Hemos comparado los algoritmos entre sí para averiguar cuál funciona mejor en cada situación

- Hemos creado un Meta-Algoritmo que es capaz de elegir y combinar los algoritmos, creando la expresión de referencia más adecuada para cada situación

12.2. Trabajo Futuro

La Generación de Expresiones de Referencia es un campo de investigación que tiene mucho que ofrecer, y este proyecto ofrece nuevas ideas y conclusiones al tema, pero debido al tiempo limitado que tenemos, lo daremos por cerrado en este punto. En caso que de se quiera retomar la investigación y continuar con el proyecto, sugerimos unas ideas que sería interesante poder tratar.

En primer lugar, nos gustaría comprobar cómo influiría en las respuestas de los usuarios tener personajes de mayor calidad. Esto permitiría apreciar y diferenciar detalles como la barba, la tonalidad de los colores o los dibujos en la ropa.

Nos gustaría añadir objetos como portátiles, comida, mochilas u objetos especiales para los personajes, como un gorro o una corbata. Éstos últimos se podrían incluir en el fichero XML como objetos *especiales*.

Sería interesante añadir personajes de diferentes alturas, porque podría ser bastante descriptivo en algunas situaciones mencionar si alguien es más alto o bajo que el resto de los personajes. Lo mismo se podría aplicar a su peso.

En la primera encuesta se reflejó que había un personaje que se podía identificar muy fácilmente por su *tipo* (*"camarera"*). En nuestra aplicación solamente usamos *tipos* distintos para referirnos a la raza (*"chico negro"*), y cuando una exprresión de referencia usaba solamente el (*tipo*), no era muy útil para los usuarios. Sería interesante poder comprobar si esto cambiaría al usar tipo más distintivos, como (*"camarera"*) o (*"cocinero"*).

En cuanto a la forma de interactuar con la aplicación, sería interesante implementar un mecanismo que permita simular una conversación, como sugerimos en las conclusiones de la última encuesta. De esta forma, se podría empezar con los atriburos más descriptivos e ir añadiendo más información a la descripción hasta que el usuario acertase con la persona que se desea. Esto podría significar implementar algoritmos nuevos y considerar los atributos en un orden distinto. Un ejemplo del resultado que se podría obtener es:

- ¿Ves al chico que está en la cola para pedir la comida?
- ¿Cuál de todos?
- El que va de azul.
- ¿El de azul oscuro?
- No, el de azul claro.

Nos gustaría comprobar si combinar en una misma descripción personas cercanas y objetos cercanos podría mejorar los resultados.

Otra cosa interesante de explorar serían los grupos de personas. En vez de considerar simplemente las personas individuales, podríamos mencionar que una persona está sentada con un determinado grupo de gente.

Appendix A

Instructions for the generation of scenes

A.1. Modifying the Scenes

This application has several different scenes. The initial scene (Main) is the first to run and it generates a few scenes (Language, Gender, Age) to gather personal data. Next, a scene appears (Curtain) that shows the description for the person that has to be found (the referent) and after that, the respective scene with all the objects and characters will appear (1, 2, 3...). We can add as many scenes as we wish. When the application has finished, the last scene should be shown (scene -1).

In order to create more scenes to gather more information about the users, we will have to modify the LoadLevel method in the previous scene so it can now go on to display the new scene. In this new scene we must also load the scene that used to go after the previous one so that the application can run its normal course. We must also include this new scene in the build. In order to include the data in the url that communicates with the PHP file, we must add the symbol "&" if there is already information in the String, and now we can add the variable and its value. In the server, the script must be refreshed to accept this new parameter.

The Main scene contains in its hierarchy two buttons for the selection of the language. When one of them is clicked on, the Language script that it has as a component will send the information to the Info game object. Info has a script with the same name that holds all the information necessary to run the application.

In order to add a new scene we must make certain modifications. We must

add a new Vector3 to the *cameras* array with the position for the new camera in this scene. In the *scenes* array we will include the number identifying the scene (this is the same as the name of the scene). We will create an array of integers and we will include it in its respective place in *possiblePeople*. This array contains the candidates for whom a description can be generated. Next, the user must select from the scene's hierarchy in Unity the characters they wish to consider as candidates. For this they have to add those characters' numbers to the *possiblePeople* array. If we wish to allow the application to run in Spanish, we must check the descriptions that the Meta-Algorithm provides for each of the candidates, manually translate them, and add them to an array. In the Resources folder we have to include an XML file that defines the characters' attributes. Now the application will be ready to run. If the furniture in the scene needs to be modified, we recommend the user to copy one of the existing scenes and modify it.

A.2. Architecture and execution flow

The scenes are stored in a folder, just like the scripts, the materials and the resources. All of the different algorithms are in separate files.

We will now explain the execution flow of the application. First, the Main scene is loaded. It loads the language selection buttons and the Info object, which runs the Start method of the Info script. In this script, the necessary values are initialised and the application waits in the Update method until the user selects a language. Next, the parameters for the url are created. It then loads the Gender scene and receives the user's data. The buttons are the ones responsible for sending the information to the url and loading the next scene, Age. Age works the same way as the Gender scene, and then loads the Curtain scene.

The Curtain scene has a new game object, *world*, a series of Transforms that represent the furniture objects (identified by different tags), and the *Start* button which will allow the user to view the scene. The world object contains a script that will be run every time this scene is played. At this moment, the variables for the generation of the referring expressions are prepared. These include the lists of objects and people, and their lists of information. Next, the objects in the scene are initialised and their attributes are added to them. Now it is the characters' turn. The ParsePeopleXML method in the SetCharacterPostures script will return a list of game objects that represent the characters. Here, the DTD can be invoked to check if the XML file is correct, although in the latest version of our project we do not do this. Now the XML file can be loaded. We look through each node, gathering each character's information and creating an object that will have

all these details inside a *struct*. Next, the character can be placed in the scene and the colliders are added to it. Lastly, we go back to the World script and return the list with all the characters.

The last step is to gather the details from the whole scene in order for the algorithms to work. At this point we calculate the distance in between the characters in the scene and we count how many times each value for a certain attribute appears in the scene, and add to a Counter object.

The button in the Curtain scene has a script which asks the Info object to generate a referring expression. Info takes the information in World and generates random numbers in order to choose one of the characters from the list of possible candidates. Next it runs the Meta-Algorithm to create the description.

The application waits until the *Start* button is pressed to load the scene and start timing the user. Now World will wait until a character is clicked on, find out which character this is with Ray Casting, and stop the timer. At this point the referring expression is generated, and all the information gets sent to Info, so it can be added to the url. Finally, the application sends a request to Info to change the scene. If the previous scene was not the last one, it will load the Curtain scene once again and then move onto the next scene, but if this scene was the last one, it will load the last scene (scene -1).

Now a WriteUrl script is responsible for connecting the server address with the information generated in the application, and it will send a petition via WWW to that address, ending the application flow. We must mention that the WWW petition will only run correctly if the address that tries to be reached is in the same domain as the application, it may not work if it is being run from the editor.

Appendix B

Installation guide and user manual

B.1. User manual

In order to use this application as a regular user on the website (recommended) you must open your browser and introduce the following address:

`tot.fdi.ucm.es/descripciones`

Next, if Unity Web Player is installed in your computer you must give it permission to launch the application. If you do not have it, you must install it manually or automatically. The next step will be to choose the language for the program (both Spanish and English are supported) and introduce your gender and age by pressing the buttons on the screen (this information will only be used for statistical purposes).

Before pressing the *Start* button make sure you read the description at the top of the screen. This is the person you will have to find in order to correctly complete all the scenes. When you are ready, press the button and you will be able to see the scene on the screen. Keep in mind that the description will remain at the top of the screen while you are in that scene, so there is no need to worry about memorising it. Once you think you have found the correct person, simply click on them and wait for the next scene to load. The process will be repeated until all the scenes have been shown, although you can close the browser at any point and if you have clicked on any characters, the information gathered so far will have been saved. Once you have found all the characters, a Thank you message will appear on your screen and the information will be saved automatically. You are now free to close your browser.

In order to use the application on another platform that is not the website, you must have an executable file before testing it (PC) or you must install it on your device (smartphone or games console). To close the application simply click on the screen.

B.2. Developer setup

You can use Unity 4.5 (the most recent version) to open and import the project, but in this case Unity will have to refresh the files, since it can also be used by older versions. The project can be compiled by clicking on File, Build Settings, then selecting the platform on which you wish to test it (Web Player, PC, MAC, Linux, iOS, Android, BlackBerry, Windows Store Apps, Windows Phone 8, Google Native Client, XBOX360, PS3, Wii or Oculus Rift if you have installed its plug-in) and clicking on *build*.

In order to upload the Web Player version, the XML files and unity3d files must be in the same folder as the PHP script that holds the data. The script must be able to access the file in which you wish to store the data.

Bibliography

Dale, R. Cooking up referring expressions. *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics (ACL)*, 1989a.

Dale, R. Generating referring expressions: Constructing descriptions in a domain of objects and processes. *The MIT Press, Cambridge, MA*, 1989b.

Grice, P. Logic and conversation. *Peter Cole and Jeffrey L. Morgan, editors, Syntax and Semantics, Vol. 3: Speech Acts. Academic Press, New York*, 1975.

Hervás, R. *Referring Expressions and Rhetorical Figures for Entity Distinction and Description in Automatically Generated Discourses*. Tesis Doctoral, 2009.

Horacek, H. A new algorithm for generating referring expressions. *Proceedings of the 12th European Conference on Artificial Intelligence*, 1996.

Kelleher, J. and Kruijff, G.-J. Incremental generation of spatial referring expressions in situated dialog. *Proceedings of the 21st International Conference on Computational Linguistics (COLING) and 44th Annual Meeting of the Association for Computational Linguistics (ACL), Sydney*, 2006.

Krahmer, E. and Theune, M. Efficient context-sensitive generation of descriptions in context. *Kees van Deemter and Rodger Kibble, editors, Information Sharing: Givenness and Newness in Language Processing. CSLI Publications, Stanford, CA*, 2002.

Krahmer, E. and Van Deemter, K. Computational Generation of Referring Expressions: A Survey. *Computational Linguistics*, vol. 38(1), páginas 173–218, 2012.

Lane, L. W., Groisman, M. and Ferreira, V. S. Don't talk about pink elephants! speakers' control over leaking private information during language production. *Psychological Science*, 2006.

Olson, D. R. Language and thought: Aspects of a cognitive theory of semantics. *Psychological Review*, 1970.

PECHMANN, T. Incremental speech production and referential overspecification. *Linguistics*, 1989.

REITER, E. and DALE, R. A fast algorithm for the generation of referring expressions. *Proceedings of the 14th International Conference on Computational Linguistics (COLING), Nantes*, 1992.

REITER, E. and DALE, R. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive Science*, 1995.

REITER, E. and DALE, R. Building natural language generation systems. *Cambridge University Press, UK*, 2000.

SONNENSCHEIN, S. The effect of redundant communication on listeners: Why different types may have different effects. *Journal of Psycholinguistic Researchs*, 1984.

TURNER, R., SOMAYAJULU, S. and REITER, E. Generating approximate geographic descriptions. *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG), Athens*, 2009.

VIETHEN, J. and DALE, R. The use of spatial relations in referring expressions. *Proceedings of the 5th International Natural Language Generation Conference (INLG), Salt Fork, OH*, 2008.

WINOGRAD, T. Understanding natural language. *Academic Press, New York*, 1972.