Perspective-Based Referring Expression Generation in a 3D Environment

Final Project for the Degree in Software Engineering

Ricardo de la Rosa Vivas
Daniel Ruiz Manero

Directed by

Raquel Hervás Ballesteros
Gonzalo Méndez Pozo

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

Madrid, 15 de Junio de 2015

Perspective-Based Referring Expression Generation in a 3D Environment

Final Project

Ricardo de la Rosa Vivas
Daniel Ruiz Manero

Directed by

Raquel Hervás Ballesteros
Gonzalo Méndez Pozo

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

Madrid, 15 de Junio de 2015

Everything has its beginning. But it doesn't at "one".
It starts a long before that... in chaos.
The world is born... from zero.

Big Boss (Metal Gear Solid 4)

Agradecimientos

Ante todo, es de agradecer a Raquel y Gonzalo por guiarnos durante el desarrollo de todo el proyecto. También por habernos dedicado mucho tiempo y atención a nuestro trabajo.

También debemos agradecer a la familia, amigos y demás conocidos su constante apoyo y comprensión.

Por último, agradecer a las personas que hayan realizado estudios que compartan el ámbito de trabajo de este proyecto.

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.

Ricardo de la Rosa Vivas Daniel Ruiz Manero

Resumen

Con la capacidad que tiene el ser humano para comunicarse verbalmente surgen una serie de problemas que no tienen precedentes en otros animales. Debido a las diferentes costumbres, orígines, sexo, raza y muchas otras razones, la forma de expresarse y analizar las situaciones pueden ser muy distintas de una persona a otra.

Este proyecto se basa en un trabajo anterior (Rabadán and Rodríguez, 2014), en el cual se exploró como las personas realizan descripciones para referirse a alguien en concreto y se creó un meta-algoritmo capaz de generar una descripción de una persona en un entorno 3D digital. No obstante, este meta-algoritmo poseía algunas limitaciones ya que se habia probado en situaciones muy concretas en donde, por ejemplo, la persona que se quiere describir estaba siempre a la vista del usuario y el campo de visión del usuario era fijo y no cambiaba.

En este proyecto se investigan como las propiedades espaciales y las diferentes perspectivas visuales de los interlocutores pueden afectar a la composición de las descripciones usadas para referirse a otra persona.

Teniendo como base el algoritmo del anterior proyecto, se ha creado un nuevo meta-algoritmo capaz de generar descripciones en circunstancias más realistas. Debido a que el funcionamiento del algoritmo anterior estaba condicionado por factores muy poco realistas, se busca que el nuevo meta-algoritmo genere descripciones claras y entendibles para seres humanos y que además esté preparado para generar descripciones en situaciones más parecidas a la vida real.

Con el motor gráfico de Unity 3D se ha hecho uso de la habitación cerrada con personajes que se empleó en el proyecto anterior para hacer las pruebas del nuevo meta-algoritmo. No obstante, se le han añadido personajes aleatorios y la habilidad de moverse a través del entorno. De esta manera, se ha elaborado un entorno 3D en donde las distintas situaciones en las que se puede probar este nuevo meta-algoritmo son mayores.

La eficacia de este nuevo meta-algoritmo se ha puesto a prueba mediante una serie de encuestas que miden el desempeño de éste a través de usuarios reales. Estos usuarios leen las descripciones generadas y buscan al personaje que se les describe. De esta manera, se pudo comprobar que las descrip-

ciones eran lo suficientemente claras para que cualquier persona las pudiera entender.

Como consecuencia de este proyecto, se ha avanzado en el campo de Generación de Expresiones de Referencia con la creación de un meta-algoritmo con un alto índice de éxito, capaz de cambiar en tiempo real y tener en cuenta las posiciones relativas del receptor de la descripción. Así, las descripciones generadas son más realistas que en el proyecto anterior.

Abstract

With the ability of communicating verbally that human beings have, a series of problems arises that have no precedents in other animals. Due to differences in customs, origins, sex, race and many others, the way humans express themselves and analize situations can be very different from one person to another.

This project is based on a previous work (Rabadán and Rodríguez, 2014), where it was researched the way in which people create descriptions to refer to someone in particular and a meta-algorithm was created, capable of generating a description of a person in digital 3D environment. Nevertheless, this meta-algorithm had limitations because it was tested in very specific ways, for example, having the person that is going to be described always in the user's field of view, and having the user's field of view fixed and unmovable.

In this project is investigated how spatial properties and the different points of view of the interlocutors affect the composition of descriptions used when referring to another person.

Having as a base the algorithm of the previous project, a new meta-algorithm has been created capable of the generation of descriptions in more realistic circumstances. The performance of the previous algorithm was conditioned by not very realistic factors, but the new meta-algorihtm aims to generate clear and understandable descriptions for human beings and also aims to be prepared to generate descriptions in situations more like real life.

Using the graphic engine Unity 3D, the previous closed room with characters was employed for testing the new meta-algorithm. Nevertheless, the ability to move around the environment and new random characters were added. This way, a 3D environment was created where the different situations the new meta-algorithm can be tested against increases.

The effectiveness of the new meta-algoritm was put to the test by a series of surveys that measure the performace through real life users. These users read the generated drescriptions and look for the character that was described. This way, it was possible to verify that the descriptions were clear enough for any person to understand.

As a consecuence of this project, an advance in the field of Generation of Referring Expressions was made with the creation of a meta-algorithm with

a high success rate, capable of adapting in real time and taking into account the relative positions of the description's receptor. This way, the generated descriptions are more realistic than in the previous project.

Palabras Clave

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

Keywords

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

Contents

List of Figures

Capítulo 1

Introducción

"Stay a while, and listen!"

Deckard Cain (Diablo)

Una de las grandes capacidades que tenemos como seres humanos es la habilidad de comunicarnos verbalmente. Esto ha permitido que evolucionemos y nos desarrollemos más allá de lo imaginable. Entre todos los posibles temas de los que podemos hablar en nuestro día a día, con mucha frecuencia está presente la necesidad de referirnos a otra persona.

En estos casos, una de las cosas más importante es tener claro sobre quién se está hablando exactamente. Esto puede llegar a ser un duro trabajo debido a las diferentes perspectivas de los interlocutores, ya sean físicas o psicológicas. Por ello es preciso plantearse cuales son los métodos y características más efectivas para identificar a una persona con rapidez y claridad.

Para alcanzar la identificación adecuada de una persona tenemos a nuestra disposición muchas característcas de ésta que se pueden utilizar, desde sus rasgos físicos y psicológicos, hasta su posición en el espacio y en el tiempo.

Este trabajo se enfoca en probar cómo las perspectivas, rasgos físicos y espaciales afectan a la identificación de una persona, y cuales de estas características son las más efectivas a la hora de identificar a una persona.

1.1 Motivación

Poder generar descripciones de personas adecuadamente y tener un entendimiento de qué patrones, palabras y expresiones utilizamos para referirnos a otras personas es importante para el desarrollo de aplicaciones inteligentes. Comprendiendo mejor este tipo de comunicación somos capaces de crear aplicaciones para ayudar a personas con discapacidades visuales o incluso avanzar en el campo de la realidad aumentada.

Se cree que el uso de la información espacial de la persona es tan crucial como las características físicas de las personas. Descripciones como "*El hombre con camisa blanca*", pueden ser más descriptivas y ayudar a una identifación más rapida teniendo en cuenta posiciones con respecto a la persona que escucha la descripción: "*El hombre que está a tu derecha con camisa blanca*".

Este trabajo está apoyado en el trabajo anterior (Rabadán and Rodríguez, 2014) "Generating Referring Expressions in a 3D Environment"Los objetivos de este trabajo fueron:

- Identificar los detalles que son importantes para la gente en las descripciones.
- Crear distintos algoritmos que generen expresiones de referencia precisas.
- Combinar estos algoritmos para crear un meta-algoritmo que ofrezca la descripción más apropiada en cada situación.
- Crear un entorno 3D para probar los algoritmos.

Este trabajo cumplió sus objetivos y logró identificar los detalles que son importantes para las personas, crear un entorno 3D y lo más importante, desarrollar un meta-algoritmo capaz de componer una descripción adecuada para la situación.

No obstante, se cree que este meta-algoritmo, condicionado por muchos factores que a la hora de aplicarlos de una forma realista en el día a día, sufre algunas deficiencias.

Las pruebas realizadas para comprobar la eficiencia del algoritmo fueron en ambientes controlados, en donde el receptor de la descripción no se podía mover y tenía una visión fija del escenario. Además, las posibles personas a ser descritas estaban todas a la vista del recpetor.

El principal objetivo del trabajo que se presenta a continuación es ampliar y mejorar este meta-algoirmo para tener en cuenta posiciones absolutas y relativas, tanto de la persona descrita, como de la persona que recibe la información de las descripciones. Esto acerca más al meta-algoritmo a una situación de la vida real.

Con este proyecto se espera contribuir en el campo de la Generación de Expresiones de Referencia, que es un subcampo dentro de la Generación de Lenguaje Natural.

1.2 Objetivos

A continuación se enumeran los objetivos específicos de este trabajo:

- Crear un entorno 3D de generación aleatoria para poner a prueba el algoritmo del generador de descripciones y tener acceso a diversos entornos de pruebas dinámicos.

- Crear las personas a ser descritas (personajes) de forma aleatoria, con el objetivo de aumentar las posiblidades de descripciones.

- Permitir a la persona que lee las descripciones (usuario) caminar por el entorno, para que pueda acercarse y alejarse de los personajes.

- Modificar el algoritmo para que genere descripciones que tengan en cuenta la posición del personaje descrito y del usuario.

- Modificar el algoritmo para que cambie en tiempo real la descripción si es necesario.

1.3 Método de trabajo

El trabajo se realizó en dos etapas. De esta manera se pudo tener en cuenta los fallos y anotaciones tomadas en la primera etapa e incorporarlas en la segunda etapa.

La primera etapa se centró en cumplir los primeros objetivos que estaban dedicados al entorno en donde el usuario realiza las encuestas. Se añadieron personajes aleatorios y se permitió al usuario moverse por la escena. El objetivo de realizar la primera encuestra con estos primeros cambios era comprobar la eficiencia del meta-algoritmo anterior en situaciones más parecidas a la vida real.

Una vez recopilada la información obtenida por la primera encuesta, en la segunda etapa se desarrolló un meta-algoritmo que tenía en cuenta este cambio en perspectiva por parte del usuario y era más parecido a situaciones de la vida real. Con estos cambios el meta-algoritmo obtuvo unos resultados mejores en la segunda encuesta, demostrando así las mejoras del meta-algoritmo.

1.4 Estructura del documento

En el Capítulo 1 se presenta el proyecto con una introducción y se habla sobre los objetivos del mismo, el método de trabajo y la estructura de este documento, tanto en Español como en Inglés.

En el Capítulo 2 se empieza por hacer un estudio del estado actual del campo en el que se trabajó, la Generación de Expresiones de Referencia. Tener claro sobre qué trabajos y estudios se puede apoyar este proyecto es crucial para realizar un trabajo satisfactorio. Además se habla sobre las

tecnologías utilizadas y en qué tecnologías se apoya este trabajo para poder realizarse.

En el Capítulo 3 se detallan los pasos y acciones realizadas para cumplir con el objetivo de mejora en el entorno 3D. Estas primeras mejoras incluyen la inclusión de personajes aleatorios, la implementación de la navegación del usuario, y todos los problemas que esto conllevó.

En el Capítulo 4 se indican los detalles de la primera encuesta. En este capítulo se explica el objetivo de esta encuesta y se recogen las herramientas preparadas para el análisis de la misma. Además se explican los resultados obtenidos en la encuesta y las implicaciones que tuvieron a la hora de cambiar y mejorar el meta-algoritmo.

En el Capítulo 5 se habla sobre los cambios realizados al meta-algoritmo, el objetivo y la preparación seguida para realizar la segunda encuestra.

En el Capítulo 6 se analizan los resultados de la segunda encuesta, la cual incluye el meta-algoritmo modificado y mejorado.

En el Capítulo 7 se detalla cómo se ha dividido la carga de trabajo entre los miembros del equipo.

Por último en el Capítulo 8 se habla sobre los resultados y las conclusiones obtenidas con la realización de este proyecto, tanto en Español como en Inglés.

Chapter 1

Introduction

"Hey! Listen!"

Navi (Ocarina of Time)

One of the great abilities that we have as human beings is the ability to communicate verbally. This has allowed us to evolve and develop more that we can ever imagine. Among all the possible subjects we can talk about, probably the most common to talk about other people.

In this cases, one important thing to have in mind is to have a clear understanding of who is we are talking about. This can be a hard job due to the difference in perspective of the interlocutors and these differences can be physical or psychological. This is why it is necessary to ask ourselves what are the methods and characteristics that are more effective to identify a person with swiftness and clarity.

To identify a person accurately we have at our disposal a lot of characteristics that can be used. From their physical and psychological traits, to their position in space and time.

This project aims to prove how perspective, physicals traits and spatial position affects the identification of a person, and which of this characteristics are the most effective to identify a person.

1.1 Motivation

Generating good descriptions of people and having an understanding of patterns, words and expressions used when referring to another person, is crucial to further develop the creation of intelligent software. By understanding this type of communication we are capable of creating applications that can help people with a visual disability or even advance in the field of augmented reality.

The use of spatial information is as important as the physical characteristics of a person. Descriptions as "The man in the white shirt" can be more descriptive and helpul identifying the person faster by using the perspectives: "The man to your right with a white shirt".

This project is based upon the previous work: (Rabadán and Rodríguez, 2014) "Generating Referring Expressions in a 3D Environment". The objectives of this project were to:

- Identify the details that are important to people in their descriptions.
- Create different algorithms that generate accurate referring expressions.
- Merge these algorithms together to create a meta-algorithm that offers the most appropriate description for the situation
- Build a 3D environment to test the algorithms in.

This project completed its objectives, found the details that are important to people, created a 3D environment and most important, created a meta-algorithm capable of creating an appropiate description for the situation.

Nonetheless, this meta-algorithm is conditioned by many factors that when applying it to real life situation present some deficiencies.

The tests made to test the algorithm's efficiency were made in a controlled environment, where the receptor of the description could not move and had a fixed position of the scene. Also, the possible people that could be described were all in the field of view of the receptor.

The main goal of the project here presented is to improve this meta-algorithm by making it take into account absolute and relative positions of the described person, and of the person receiving the description. This way, the meta-algorithm gets closer to function in a real life situation.

This project hopes to contribute to the field of the Generation of Referring Expressions, which is a subfield inside Generation of Natural Lenguage.

1.2 Objectives

The specific objectives of this project were to:

- Create a random generated 3D environment to test the description generator's algorithm and to have access to an array of dynamic test environments.
- Make the creation of described persons (characters) random. This way the possibilities of different descriptions increases.

- Allow the person who read the descriptions (user) to walk around the environment. This way he or she can approach or distant themselves from the characters.
- Modify the algorithm so it can generate descriptions that take into account the positions of the described character and of the user.
- Modify the algorithm so it changes its description in real time if necessary.

1.3 Work Method

The project was done in two stages. This way the flaws and annotations from the previous stage could be taken into account when preparing for the next stage.

The first stage focused on the objectives dedicated to the 3D environment where the user does the survey. Random characters were added and the ability to move was implemented. The objective of the first survey with these changes added was to test the previous meta-algorithm's efficiency in situations closer to real life.

With all the information from the first survey, in the second stage an improved algorithm was created that took into account the user's perspective, and was more prepare for real life situations. With these changes, the meta-algorithm gathered better results in the second survey.

1.4 Document Structure

In Chapter 1 an introduction of this project is given, followed by the specific objectives, work method and the structure of this document. This is done both in Spanish and in English.

Chapter 2 begins by making a study of the current state of the field of this project, Generation of Referring Expressions. Understanding the work of others in this field is key to make a great project. In addition, this chapter has information on the technologies used in the making of this project.

Chapter 3 details the steps made to fulfill the 3D environment improvement objectives. These first improvements included the addition of random characters, the implementation of the user's navigation and all the problems this brought.

Chapter 4 has details of the first survey. In this chapter, an explanation of the objectives is given, as well as the tools used to prepare the analysis of the results. Additionally, this chapter contains the analysis of the results given by the survey, and the implications that these results had when changing the algorithm.

Chapter 5 details the changes made to the algorithm, the objectives and the preparation for the second survey. This survey main objective was to test the improvements done to the algorithm.

Chapter 6 contains the analysis of the result of the second survey, which has the improved and modified algorithm.

Chapter 7 details how the workload was divided between the members of the team.

Lasly, in Chapter 8 the results and conclusions obtained from this project are presented. This is done both in Spanish and in English.

Chapter 2

Related Work

"Beware of he who would deny you access to information, for in his heart he dreams himself your master."

Commissioner Pravin Lal (Alpha Centauri)

2.1 State of the art

A referring expression is a natural language sentence which identifies univocally a partial or full entity. The Generation of Referring Expresions is the process of creating referring expressions in order to make an individual able to identify the descibed entity as easily as possible (Winograd, 1972; Hervás, 2009). It is a part of the Natural Languaje Generation (NLG) which studies the generation of natural language. The described entity can be either a person or an object or anything that can be described with words.

A referring expression is a natural language sentence which identifies univocally a partial or full entity. The Generation of Referring Expresions is the process of creating referring expressions in order to make an individual able to identify the descibed entity as easily as possible (Winograd, 1972). It is a part of the Natural Languaje Generation (NLG) which studies the generation of natural language. The described entity can be either a person or an object or anything that can be described with words.

In order to be a successful generated referring expression, it has to satisfy several criterias (Reiter and Dale, 1995):

- **Referential Success:** The expression must refer to the entity univocally. That means that the expression must contain attributes that make the entity unique in an scene. No misunderstanding should be expressed in a generated referring expression.

- **Ease of Comprehension:** The expression must be quickly and easily understandable for any kind of reader. Understandable and well known attributes must appear in the expression in order to achieve this criteria.

- **Computational Complexity:** The generation of the referring expression must be fastly generated. The Time Complexity of the algorithm must be as good as possible.

There are several algorithms in the literature which implement the actions described beforehand.

- **Full Brevity** (Dale, 1989a): This algorithm minimises the number of attributes or properties that appear in a generated description. It is not an algorithm but a collection of lesser algorithms which makes this one. It is computationally intratable because in the worst case the time grows exponentially when trying to find a minimal description.

- **Greedy** (Dale, 1989a,b): It selects the attributes which are true rather than searching exhaustively. This is the fastest one. It does not always generate minimal descriptions because an attribute that removes several attributes when added, it might not remove future attribute additions. This one is useful because people often prefer some attributes over others (Pechmann, 1989).

- **Incremental** (Reiter and Dale, 1992, 1995): It selects attributes one by one and stops when the combination of those attributes is considered to be successfully descriptive.

- **Relational Algorithms** (Horacek, 1996; Krahmer and Theune, 2002): They consider the surrounding objects or elements in order to be used in a description. For example, *the man near the window.*

2.2 Previous Work

This project is a continuation from the project previously developed and written by (Rabadán and Rodríguez, 2014).

The objectives of this project was to identify the details that are important for most people when making a description and to create an algorithm that generated the most apporpiate description depending on the situation

This project was guided by a three surveys. People were asked about certain aspects about character descriptions before the creation or the modification of the Unity project. It was done this way so the code was correctly written even before the start.

They began the project by testing the behavior of the user. The users were given a set of photographs of real people of the canteen of our University and they were asked to identify some people. The identification was done by using the descriptions of the people given for the test. The user had to jot down the description they thought was proper to be used in order to describe the referred person. The objective of this test was to check which attributes people use when describing other people.

The conclusion was very simple: the descriptions generated by the program must be as descriptive as people think they have to be. People had focused on color clothes, postures and the closest surroundings in order to identify the subjects. Another important element was that people focused on beards and glasses too. It was interesting because those elements are smaller and less visible than other parts of the body.

The first algorithms were created based upon the conclusions of this first survey. They were the following:

- **Exhaustive:** It generated a full and complete description of the character including all the attributes that make a person to be physically described. It could use more attributes than the ones needed.
- **Incremental:** It checked the attributes of a character based on a priority order. That priority order was the one obtained in the first survey. The priority order of the attributes up to this point was: type (profession), posture, beard, hair colour, hair length, top colour, top type, bottom colour and bottom type. Type was always used but it could be empty. If an attribute ruled out a list of distractors (people who has common attributes to the one decided to be described), that attribute was added to the description. The algorithm finished when there were no distractors or attributes. It generated realistic descriptions.
- **Greedy:** It chose an attribute depending on the situation. It chose the attribute that was the most distinguishing one for the remaining list of distractors. Type attribute was added too. If there was more than one distinguishing attribute, a priority order would determine which one had to be processed (the same priority order than in the Incremental algorithm). It finished when there were no more attributes or distractors and generated realistic descriptions too.
- **Nearby People:** It was a relational algorithm. It checked if a person was near the described person. If the person was close to the described person (less than 1.5 meters) a description would be generated for that selected person. It had to be used with a non-relational algorithm: Exhaustive, Incremental or Greedy.
- **Nearby Objects:** It was a relational algorithm and must be used with a non-relational algorithm too. It worked similar to the one ex-

plained beforehand. It checked the distance from the described person to near objects (objects closer than 1.5 meters) and used them in the description. If there was an object that met the proximity condition it would use the object. Otherwise, it worked as the non-relational algorithm.

In addition, they created 3D character models with persistent life cycle (the character information is stored in an XML file). The scene of the canteen was included at this point too. A scene was needed because referring expressions (descriptions) of a character can be referred to the environment too as they had concluded in the first survey.

The second test was almost similar to the first one. The main difference was that the photographs and descriptions given to the user were created using the Unity models and scene. The execution of questions is similar than before. The users were asked to rate the generated descriptions so they could consider the effectiveness of the algorithms.

The conclusions of this test were similar to the previous one too. People mostly focused on color clothes, postures and the surroundings as expected. The algorithms were working fine too.

Three main modifications were done to the algorithms at this point:

- The hair attribute was decreased in priority because few people referred to that part of the body during descriptions.
- The beards were removed too because they could not be seen from far away.
- The most important modification is the removal of the bottom part of the body when describing. People almost always use the upper part of the body when they are referring to a person.

A meta-algorithm was created at this point in order to generate the best description possible depending on the nature and the consistency of the scene. Each algorithm worked fine individually but they worked differently depending on the scene. The result was the combination of the previous selected algorithms. Not every single mentioned algorithm was included in this algorithm. We are going to discuss more about the Meta-Algorithm and the used GRE algorithms in section 2.2.1.

The last survey was focused in measuring the time the user took to find the described character. They obtained two main results:

- The time for answering the questions was reduced when the survey was getting close to the end.
- The total percent of correct hit answers was 93.4% (6.6% of error). This means the meta-algorithm worked fine.

2.2.1 Meta-Algorithm

The Meta-Algorithm is an smart combination of different GRE algorithms. By smart we mean that not every single GRE algorithm is used in this one but the ones the surveys proved to be better. Depending on the situation, the Meta-Algorithm uses an specific GRE algorithm or other. The algorithms the Meta-Algorithm is composed of are:

- Incremental
- Greedy
- Nearby People with Incremental
- Nearby Objects with Greedy

The Meta-Algorithm executes an specific algorithm depending on the data of the described person and its surrounding area. The execution flow is very simple. Nearby People with Incremental algorithm is called in the first place. If it returns exactly 1 attribute (one characteristic of the described person), that algorithm is used. The other case is when it returns several attributes. Then Greedy is executed. If it returns one attribute, Incremental algorithm is executed and used.

The last algorithm executed is Nearby Objects with Greedy Algorithm. It checks the distance between the described person and the surrounding objects. If any object is close to the described person Nearby Objects with Greedy Algorithm is used. If every object surrounding the described person is far, then Greedy Algorithm is used. We can se an example of a scene in Figure 2.1

This algorithm generates descriptions based on three factors:

- **Characteristics of the described person** and its surrounding people.
- **Proximity of the described person to objects** or elements of the scene.
- **Proximity of the described person to people** with special characteristics (special positions or clothes).

These descriptions are absolute and do not depend on the user's point of view.

2.3 Technological Background: Unity

In order to understand some sections of the implementation is necessary to know some basic concepts about the technology we are going to use during

Figure 2.1: Example of a scene of the previous project

the development of this project. This information has been taken from the Unity's online manual and scripting API (Unity-Technologies, 2015a,b). It is also necessary to explain briefly how the previous implementation was built.

Unity is a cross-platform game creation system. It can simulate 3D environments with modeled characters. We have decided to continue with this engine because the enhancements we have planned can be implemented in this system. The previous project is developed in this system as well.

The language used to implement the logic and the new algorithms is C#. You can use Javascript or Boo in order to implement behaviours too. We decide to use C# because the previous project was developed using that languaje.

It is necessary to know the basic architecture of Unity in order to understand the workflow of Unity and our improvements to the code.

2.3.1 Architecture

The architecture in Unity is based upon three simple elements: *Scenes*, *GameObjects* and *Components*.

Scenes in Unity are boards or collections of entities with certain configuration. Every Unity project must have at least one Scene in order to work out properly.

A GameObject is every single entity that can appear on a Scene. It can have a hierarchical order. That means that a GameObject can be a child of another GameObject or a GameObject can be the parent of other

GameObject (depending on the point of view).

A Component is an element that can be attached to a GameObject in order to make a GameObject have several properties or behaviours. The most basic Component is the Transform Component. That Component indicates the position, rotation and scale among other properties of the GameObject in an scene. Every GameObject can have several Components, but at least it always has a Transform Component.

The most common Components (not counting the Transform Component) that we can find are:

- **Mesh:** Collection of little triangles that makes the model.
- **Render:** Visual properties of the GameObject to be displayed.
- **Collider:** Collition detection element.
- **Rigidbody:** Physics behaviour in GameObjects (gravity and forces).
- **Script:** Logical component, indicates some behaviours over the GameObject.

Another element of Unity's Physics engine is Raycasting. We have to use this Unity's characteristic in certain improvements of our code.

2.3.2 MonoBehaviour Class

Every class / script defined in Unity Inherites from *MonoBehaviour*. It is the most basic scripting class and defines certain methods that the Unity engine is responsible to execute at certain moments. There are several interesting methods defined in this class. However, we are only going to explain the two most basic mehods:

- **void Start():** This method is called after a *GameObject* is enabled. It is most commonly used to initialize certain variables in runtime. It is called one time in the *GameObject's* lifetime. Unity guarantees us that this method is called before the Update() method.
- **void Update():** It is called each frame if the script is enabled. Most scripts that need to retrieve or adjust variables during the execution time override this method. We can use *Time.deltaTime* in order to obtain the variation of time between the execution of two consecutive frames.

2.3.3 Vector3 Class

Vector3 is a class defined in Unity's API which is used to represent spatial points and directions. It has the most common mathematical vector operations defined in it. The position variable in the *Transform* component is represented as a *Vector3*. Every *Vector3* object has three variables: x, y and z.

It is common to think that rotations use *Vector3* but it is a really big mistake. Rotations in Unity use *Quaternions*. The most common *Quaternion* used to rotate *GameObjects* is the one base on *Euler Angles*.

2.3.4 Raycasting

Raycasting is an element of the Physics engine of Unity. It allows us to shoot abstract rays from one point with an specific direction. They are really useful to determine the existence of certain objects between two points. They can be used to detect the object a user has clicked on the screen.

The properties of rays are:

- **Origin:** The main point from where the ray is going to be shot.
- **Direction:** The direction of the shot ray.
- **Distance:** The distance to the point where the ray is hitting.
- **HitInfo:** The information of the first object that collides with the ray.
- **LayerMask:** A mask that indicates that the ray can collide with certain objects.

It is important to know that rays are stopped whenever they collide with a *GameObject* that has a *Collider* attached.

Chapter 3

Dynamic Generation of Realistic Situations

"The best solution to a problem is usually the easiest one..."

GLaDOS (Portal 2)

In this chapter we are going to explain how we are going to fullfil our three first objectives described in Section 1.2.

The scene where we are going to execute the new tests or simulations was already created. That scene is known as the Canteen and is a 3D model of the canteen of our University. We want to separate the Unity models (presentation and application layer) from the logic (business layer). In order to achieve this task, we are using a multilayered architecture. The persistence (data access layer) is separated from the rest too. We use XML file to store the character data.

One of the objectives is to have every scene created at runtime and with random characteristics. This way, the potential of the algorithms will be put to the test by putting them through unexpected possibilities.

Another important matter was the implementation of a first person perspective. It has been decided to implement this new aspect because we want the user to detect a described character when this character cannot always be seen from the user's point of view. The user will have to move through the scene in order to see and select the described character.

This new perspective will make the user able to perform a close up perspective. It can be useful when the only difference between two people are their wearing clothes (for example, to distinguish a white shirt from a white t-shirt).

The last but not least improvement is the one refered to code legibility. In the beginning, we had serious problems in order to understand what the

code was used for. We are going to explain what the changes consisted of.

3.1 Dynamic Generation of Scenes

The first element that has been decided to implement is the generation of dynamic scenes.

There were three proposals in order to achieve our objectives:

1. Stop using XML to speed up the process of generating the scene and to avoid working double since the characters will be created dynamically. This was not used because by eliminating the XML files, the description algorithm will be bound to a specific engine, in this case Unity3D.

2. Creating characters dynamically, then translating them to a XML file and then instancing them in the scene. By creating characters dynamically first and then translating to XML, the same problem as in the first case arises because the creation of the XML files are bound to the engine.

3. Creating the XML dynamically and perform the creation of the characters by reading the XML. This method preserves the previous way of instantiating characters but changes the way the XML is created. This was the chosen method.

3.1.1 Random Generation of an XML file

The main idea was to have a script that, given a set of values, created the XML information corresponding to that character. These values are the possible characteristics that the algorithm reads on a character, for example, the color of the hair, the type of shirt it is wearing, etc.

These values were going to be searched in the directories where the models of the clothes were. This way, by adding a model to its corresponding directory, it will be automatically added to the possible values of the XML. Unfortunately the engine did not support this kind of directory search.

To solve this problem, the XML generator script knows what types and possibilities of each piece of clothing there are. By using a random number generator, this script is able to write multiple randomized characters and construct the XML. We can see an example of the XML content in Figure 3.1.

3.1.2 Dynamic Character Creation

Given an already built XML, a script named *PersonGenerator.cs* reads the XML and instantiate a character in the scene. Previously, all the models

```
<people>
    <!-- All people -->
  <person>
    <type>boy</type>
    <name>chico15</name>
    <sex>male</sex>
    <height>165</height>
    <hair>
      <length>short</length>
      <colour>black</colour>
    </hair>
    <beard>yes</beard>
    <clothes>
      <top>
        <type>shirt</type>
        <colour>black</colour>
      </top>
      <bottom>
        <type>3/4 pants</type>
        <colour>green</colour>
      </bottom>
    </clothes>
    <posture>posture15</posture>
    <realposture>sitting</realposture>
    <object>phone</object>
    <object>watch</object>
    <special>braid</special>
  </person>
```

Figure 3.1: Characted data stored in an XML file

were already constructed and this script only had to read the name of the models and instantiate them; now this cannot be done the same way because the characters are created dynamically. To achieve this, there were several proposals, among them it was to create the character piece by piece, meaning each part of the body and piece of clothing is instantiated separately.

Apart from being a slow process, a major engine related problem was encountered: by instantiating the parts separately each one of them had a complete full skeleton. This skeleton is used by the engine to give a model a certain posture. By having repeated skeletons for one character, creating a script that manage each skeleton separately or one that unified the skeletons into one was not an option because of the high coding problems that this would bring.

The chosen solution for this skeleton problem was to have only two characters models, one female and the other male. These characters would have all the possible clothes that are available for their gender. In this way, when instantiating the character there will be only one skeleton. Instead of cre-

ating the character from nothing and adding the clothes, the "fully clothed character" would be instantiated and have specific clothes removed.

Because the name of the clothes models inside the engine were not descriptive enough, a class named *Garment* was created to store the description name of the clothes and the engine name of the clothes. By having this class and organizing the clothes in lists by the place they are worn, the process of eliminating each piece of clothing from the characters becomes easier.

For simplicity the hair style of the character is treated like a piece of clothing. This is why when eliminating a piece of clothing, there needs to be a value that discerns between where inside the character the elimination is going to take place, in clothes or in hair. This makes it easier for the engine to find the model to be eliminated.

3.1.3 Importing New Assets

In the previous project, an order was placed to a group of 3D modelers to create characters and pieces of clothing for Unity in order to have a bigger array of possibilities when using the algorithm for describing them. These models were not ready in time to be used in the previous iteration of the project; now they are ready to be used. This would provide a more extensive pool of possibilities for descriptions. In Figure 3.2 we can see some examples of new assets.

Unfortunately, these models did not came without their flaws. Some of them had visual issues that could only be fixed by remodeling or by creating the character piece by piece. We can see in Figure 3.3 that the trunk of the character overlaps with the shirt.

In one hand, due to having no knowledge of modeling, remodeling was not an option. In the other hand, creating the character piece by piece, as mentioned before, was not viable because of the coding issues that could arise. Furthermore, the creators of these models refused to remodel the pieces of clothing.

Ultimately, a decision was made to use the piece of clothing whether or not it had visual issues. If the ones with problem were not used, the possibilities of descriptions were diminished; and by using them, the description were not altered, and their flaws were not too extensive to affect a visual recognition by the user.

3.1.4 Posture Problem

Because the previous models that were used for the descriptions were part of the 3D engine, there were not problems when translating and rotating body parts to move the character into a particular position. In this instance, because the models were created in another 3D engine and then imported to the one used, this body parts rotations came with a nonstandard numerical

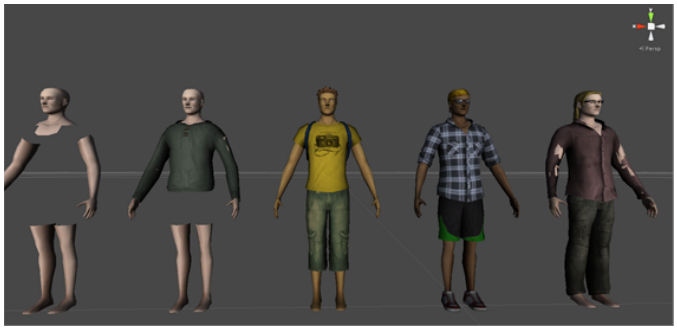

Figure 3.2: Some new clothing models

Figure 3.3: Character with visual issues

value.

For example, the rotation of a knee when in rest is (0,0,0) in X, Y and Z axis. In the models that were received, the rotations were approximately (5,171,119). This causes incorrect postures when using standard values for having a character in a sitting position, standing position, etc.

The first option was to use a 3D modeling program to change these values, but this was not a success due to an import/export problem with the 3D engine. As an alternative, the creators of the 3D models were asked to fix this issue, but they refused to remake the models. That left us with the option of changing the values of rotation manually and for all the postures. This took a long time because there were 70 posture. Nonetheless it worked as expected.

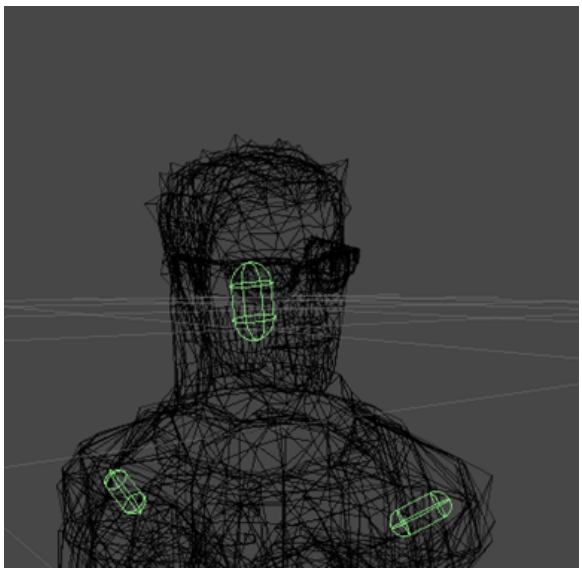

Figure 3.4: The problem of the character colliders in the new models

Due to this import and export issues, when instantiating the colliders (used by the engine to detect the users click with the mouse) these were smaller than required and had a wrong direction as can be seen in Figure 3.4. Fortunately this was not hard to fix; by increasing its size and changing manually the way the colliders were instantiated, the colliders had correct directions and sizes as can be seen in Figure 3.5.

3.2 First Person Perspective

The first person perspective is an important modification too. The main idea considering this implementation is to analyze the algorithm's behaviour in those situations in which the user is able to move over a room. That means that the described person could not be always visible from the user's point of view.

Some collateral damages or problems may appear during this implementation. We are going to discuss, analyze and solve those possible problems next.

3.2.1 Scene Adjustments

The scenes that were created in the previous project do not have colliders associated to them. The first thing was to create colliders for each of the objects of the scenes. It is necessary to put colliders in the scene elements because if there are not any colliders, the user will not be able to move over

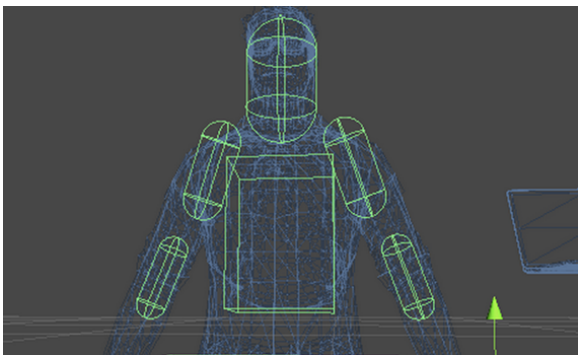

Figure 3.5: The solution for character colliders

the floor (the user will be continuosly falling) and could pass through the scene elements(like walls or tables). This can be done in Unity by selecting the component we want to have a collider and creating a mesh collider. This collider is created according to the mesh of the associated *GameObject*.

The new canteen with colliders in the tables, chairs and columns is the one used in every scene. This new canteen is copied in each scene and the previous canteen is deleted. We need these colliders on the scene so the first person character cannot go through certain elements. Another element added is the terrain, so the user can move over the floor of the cafeteria.

The next task was to integrate a first person camera in our scenes. We can find a first person camera in the Standart Assets of the Unity Store. This camera is called *FirstPersonController*.

The camera array (the array which indicates the position of each camera on the scene) is no longer needed because now the user can move the camera. We have to adjust the selected camera of the scene. The system has to select the camera that belongs to the *FirstPersonController*.

Besides the addition of the camera, this *GameObject* adds a controller with two characteristics:

1. **Movements** of the character over a terrain. It binds our point of view (camera) with the W,A,S,D Keyboard keys with the following movements: forward, leftward, backward and rightward movements respectively.

2. **Rotationary movements** for the camera. These horizontal and vertical movements are associated with the X and Y axis of the mouse respectively.

If we add this controller, the navigation through this environment is more realistic. We have to adjust the *Near Clipping Plane* to 0.1 in the Main

Camera object of the *First Person Controller*. We must do this adjustment if we do not want to see through walls if we are close to them. This problem can be seen in Figure 3.6.

The first person navigation must give the user descriptions (referring expressions) of the character he is seeking. We have concluded that we needed a simple *Heads-Up Display (HUD)* in our simulation.We decided that this indication will appear a couple of seconds on the top left of the screen and then it will disappear. If the user wants receive the indication of the searched character again, he must press the H key. The description appears on the top left part of the screen. It lasts for a couple of seconds and it disappears again.

The system retrieves the number of H keys pressed and stores the data as an array (in the *Info.cs* Script) in order to be used in future analysis. The description that is shown on the left upper part of the scene is generated when the curtain is loaded. The curtain scene (also known as black curtain scene) is the scene where the description appears for the first time per character description. The user should click on the start button in order to begin the 3D simulation. The description shown in the curtain scene is stored in the gui attribute of the *Info.cs* script. We modify the code so the description shown is the one of that mentioned script. We can see a the first prototype of this new perspective in Figure 3.7.

We have also created a timer in the curtain scene. It has been created because we think the system has to load the next scene after the user has seen the description for at least three seconds. The user will see and memorize the description for three seconds so we can be sure that our descriptions are correct. That indicated time may change in the future (due to other related facts). We have implemented this aspect but we are not really sure if this will be used in future simulations.

Room lightning has been adjusted so the navigation through the scene is comfortable and it does not affect the surveys. The simulation explains or remembers the user that he can move through the scene in the black curtain scene.

Left-click mouse detection does not work correctly. The problem is that there are two cameras in a scene. One corresponds to the camera of the first person view and the other was the one used for the previous simulations. The camera used in the previous simulations has been deleted. We have changed the tag of the *MainCamera* of the *FirstPersonController* from camera to *MainCamera* as can be seen in Figure 3.8. We have also added a left-click mouse counting in order to count how many times that button is pressed. This is used to detect if the user is randomly clicking the enviroment. When clicking any character (right or wrong) the next scene will be loaded without telling the user if he has clicked the right character.

Figure 3.6: The near clipping problem

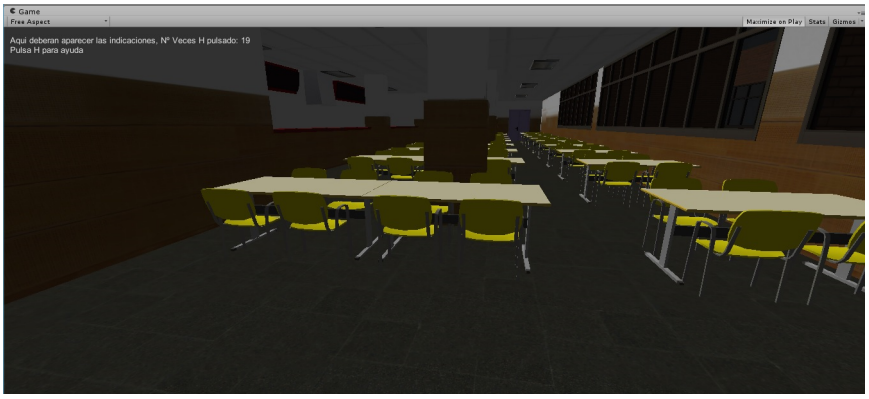

Figure 3.7: First prototype of the implementation

3.2.2 Crosshair Implementation

In the previous project, the clicked character was obtained by shooting a *Raycast* from the mouse cursor perpendicular to the screen. That implementation cannot be used when using the *FirstPersonController*. If you shoot a ray from the mouse cursor to the screen when the camera is moving, the direction of the ray can be altered because of this movement. The camera movement speed (the speed obtained when moving the mouse) and the cursor movement speed almost alway is not the same while performing a movement. This can cause the user to click a wrong character accidentally. We decided to implement a crosshair (like in FPS games) in order to cast the ray from the crosshairs center. This new implementation solves the problems related beforehand.

We created a simple black crosshair using the free and open-source graphics editor called *Gimp*. The created crosshair is the one shown in Figure 3.9.

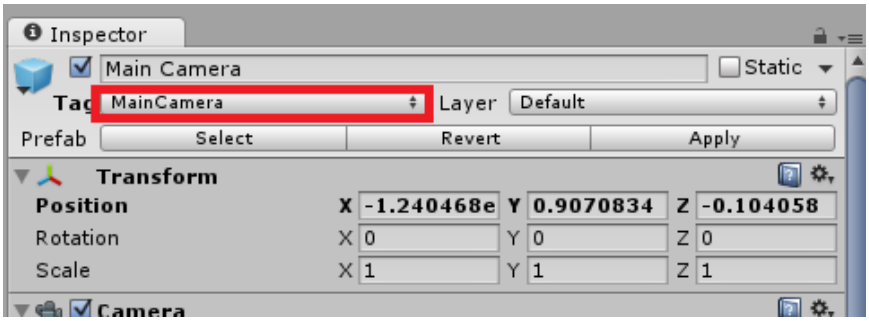

Figure 3.8: Tag selected on the *MainCamera* of the *FirstPersonController*

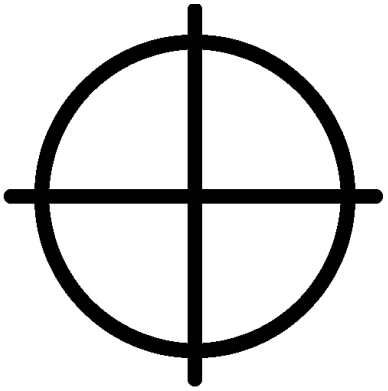

Figure 3.9: The Crosshair created using *Gimp* editor

The load and positional adjustment of this element is indicated in the *FirstPersonHUD.cs* script. We decided to make the crosshair proportional to the screen size. The crosshair width is a 2.5% the screen width. The mouse click detection is located in the *Update* method of the *World.cs* script. The mouse click detection is located in that Script because *World.cs* manage scenes loading and the meta-algorithm calls. The results of the crosshair implementation are shown in Figure 3.10.

3.3 Refactoring

We want to facilitate or improve the flexibility, comprehension and maintainment of the given code. We have decided to improvce code legibility because we have had difficulties when reading the code and executing changes. We have adjusted little fragments of the code in order to have better readability.

First, general previous functionality has not been modified. It is modified when we consider the code to be more readable. Every piece of new

Figure 3.10: The scene using the new Crosshair

code/script has been created bearing in mind future improvents or developments. That means that the new code is fully commented in order to avoid any kind of misundestanding. The previous code has been commented too. An example of new commented code is the one seen in Figure 3.11.

Some of the code readability related improvements, are the following:

- The name of the method in *Info* named *StopTimer*, has been changed to *SaveResults*. This method did stop the timer but also did a more important task that is saving the information of a particular description and if the user clicked the correct character. Also a *debug.log* has been added to check if the clicked person is the correct one or not.
- Another improvement related to this aspect is the creation of the *howTo* file in the Unity project. It has the basic indications for a developer to understand the basics of this application.

```
//TOP LEFT LABELS
//################

//The position of the top left text of the HUD
private Rect topLeftTextRect;

//The top left text of the HUD
private string topLeftText;

//CENTER LABEL
//#############

//The position of the center text of the HUD
private Rect centerTextRect;

//The center text of the HUD
private string centerText;

//Indicates how many times the help button is pressed
private int numberHelpPressed;

//The clue shown at the topleft part of the screen
private string clue;

//A reference for the info Script (data)
private Info info;

//The Crosshair
private Rect crosshairRect;

Texture crosshairTexture;
```

Figure 3.11: New commented code from the *FirstPersonHUD.cs*

Chapter 4

Preliminary Evaluation

"The right man in the wrong place can make all the difference in the world."

The G-man
(Half life 2)

4.1 Purpose of this survey

The algorithm used during this survey is the Meta-Algorithm of the original work by (Rabadán and Rodríguez, 2014). We used the Meta-Algorithm because the objective of this survey is to measure its effectiveness when the user is able to move around the place during the 3D scene simulation. It was expected that, since the algorithm describes the target person from the point of view where the user starts, when moving around the users would have problems finding persons that were not visible until the user moved, for example, characters that were behind columns.

Another aspect that was aim to put to the test was how memorable are the descriptions. If the user needs to read the description repeatedly, it may be an indication that the description is not well generated, or just too complicated to understand in just one read.

Furthermore, because the characters are now generated dynamically, we wanted to test the algorithm performance when having random and unexpected circumstances, for example, a lot of people generated in one side of the scene and in the other side almost none. We can see some generated scene examples in Section 4.5.

4.2 Survey Instructions

Because the survey is now more complicated , an introductory message was displayed when starting the survey to make sure the user knows that the

ability to move is now an option and how to do so. This message is merely informative and is not taken into account when analyzing the results.

The ability to move around the scene may confuse some users that are not used to FPS *(first person shooter)* controls; this was taken into account by asking the person if they have ever used this way of moving around the scene. With this information, we can take into account the time some users may spend getting used to the controls instead of searching for the described character.

To measure how memorable the descriptions are, the descriptions disappear after a certain amount of time and a key was implemented so the user can make the description appear again for a brief period of time. This way, the amount of times the user pressed the description can be contrasted with the description's complexity. In addition to this key, a variable to measure the numbers of clicks done by the user was implemented with the objective to validate the survey and to spot users that click randomly.

Although there is the capability to make all surveys random and different, in the interest of statistics and cleaner results, we choose them from a number of randomly created scenarios. Those that were better suited to measure the algorithms capabilities were selected. This way, all the users were confronted with the same scenarios and the results were more comparable.

The survey instructions are in Spanish because surveyed people main language was Spanish, this way we made sure that users understood the instructions. However, the survey is completely in English because the generated sentences for the descriptions have an English structure. We can see this instructions in Figure 4.1.

Figure 4.1: Instructions of the first survey

4.3 Capturing survey results

Once the user is done with the survey, the information gathered is stored in a simple text file. It is generated on the server whenever a user completes the survey, not during the survey. Because this information did not came out in an organised fashion, it was decided to changed the format of the information stored inside that file to increase the legibility and make it easier to process.

There are two reasons why the changed of the format was neccesary:

1. To make the data file more readable for human analyzers.
2. To prepare the file to be the input for the *Survey Analyzer Application*.

The differences between the previous format and the new format are explained in the following section.

4.3.1 Previous Format

Each line represented a completed survey. We can find language, gender and age data at the beginning of the line corresponding to the user data. The information following the aforementioned data contains the scene information.

The information of the scene is:

1. The number of the Scene.
2. The algorithm used for the description of the searched character.
3. The description of the character the user has to find.
4. The algorithm used for the description of the clicked character.
5. The description of the character the user selected.
6. Time required for the user to find the character.
7. If the user had found the searched character.

By having all the data from each user in a single line, it was difficult to read and to make conclusions based on this information.

4.3.2 New Format

In the new format a group lines represents a survey and each scene is represented in one line. For each group if lines, the first line represents the data of the user. We can find language (always English because the descriptions

are generated in English), gender, age and if the user has experience in *First Person* games.

Each one of the next lines is an scene. We have prepared 10 scenes for this survey. The data stored for each scene is almost the same as the previous one but with small changes.

Between the descriptions of the clicked character and the time that the user took to find the character we can find:

1. The number of clicks pressed (almost always 0 if the user has not clicked the stage by error)

2. The number of H pressed.

These simple changes made the analysis quite simpler and made easier the process of arriving to conlusions.

4.4 Survey Analyzer Application

We have decided to build an automatic *Survey Analyzer*. It analyzes simple information like the algebraic means of some data and correct answers per scene. The language used to build this application is *Java*. It has been decided this way, so we can use it in various platforms (Windows, Linux...).

Survey Analysis gets simpler with the development of this application, and this analyzer can be useful in future analysis. In addition this analyzer can be upgraded to process the information in different ways to acquired specific data.

We can see the analysis of the output data of the application in Section 4.5.

4.5 First Survey Result Analysis

Due to the survey being complicated and long, a total of twenty seven people did the first survey. Most of them were in their twenties and a few in their thirties. Only 15% were females. The survey was done by more people but they did not finish it completely, so the registered information was not stored. Fortunately we were present in some of this cases and managed to receive feedback from the users.

We present the individual results for each scene. Each scene was selected specifically to test the algorithm performance when the user is able to move and when the described person is not visible. In each individual result the first picture corresponds to how the user starts in the scene without any movement, and the second is where the described person is.

To clarify who the person is, a red circle has been drawn around the described person. Among the statistics that the Survey Analyzer gives, Mean H Pressed refers to the average number of times the users pressed the button H, to make the description appear, per scene. Also Mean Time is the average time all users took to complete a certain scene.

4.5.1 Scene 1

As can been seen in Figure 4.2, the described person is behind a character and he is far, so the user has to move inside the scene. This way the user can get accustomed to the controls and realize that is mandatory to move to find certain people. Once the user moved, it was not difficult to find the described person. In Figure 4.3 we can see the target from a closer point of view.

The data obtained from the experiments was the following:

- **Used Algorithm:** NearbyObjectsWithGreedy
- **Shown Description:** The boy in the black shirt sitting near the window.
- **Number of Correct Clicks:** 23/27
- **Mean H Pressed:** 0,15 times
- **Mean Time:** 3,79 seconds

This scene was key for users that were not accustomed to this type of controls. The users with higher times in this scene are the ones that were not used to the controls. Despite of this, this scene had a high number of correct clicks: 23 out of 27 people found the target person.

Figure 4.2: Survey 1 scene 1 starting position

Figure 4.3: Survey 1 scene 1 described person

4.5.2 Scene 2

We can see in Figure 4.4 that the target person is in the field of view of the user. The main purpose of this scene was to test the algorithm performance when having many characters that look like the target person and were close to the target. As can be seen in Figure 4.5 the target character is next to many people sitting, with blue shirt and with black hair. This caused many people to get confused and choose the wrong person.

The data obtained from the experiments was the following:

- **Used Algorithm:** NearbyObjectsWithGreedy
- **Shown Description:** The boy in the blue t-shirt sitting with black hair near the window.
- **Number of Correct Clicks:** 18/27
- **Mean H Pressed:** 0,00 times
- **Mean Time:** 4,88 seconds

It is also worth noting that the key H was pressed an average of 0 times. This may be the reason why this scene got a lower score, 18 out of 27, than the previous one. Because the description was fairly easy, no one felt that it was necessary to read the description again, causing the users to make a rapid decision and choose one of the character that look like the target.

We believe that this type of situation can be solved by avoiding the use of a descriptor that many characters, nearby the target, have; in this case it is the color blue.

Figure 4.4: Survey 1 scene 2 starting position

Figure 4.5: Survey 1 scene 2 described person

4.5.3 Scene 3

As can be seen in Figure 4.6 this is the first scene where the target character is behind an object. Fortunately the meta-algorithm chose the NearbyPeople algorithm, making this scene easier by describing the girl in the yellow tank top (yellow circle) that is visible from the start and is next to the target.

The data obtained from the experiments was the following:

- **Used Algorithm:** NearbyPeopleDescriptionWithIncremental
- **Shown Description:** The girl in the blue sweater with black hair. She is near, next to the girl in the yellow tank top.
- **Number of Correct Clicks:** 24/27
- **Mean H Pressed:** 0,15 times
- **Mean Time:** 3,01 seconds

We believe that a lot of people got this scene correct, 24 out of 27, because of having the only girl with a yellow tank top in the description, even when she was not the target. In Figure 4.7 we have a closer look of the target person.

Figure 4.6: Survey 1 scene 3 starting position

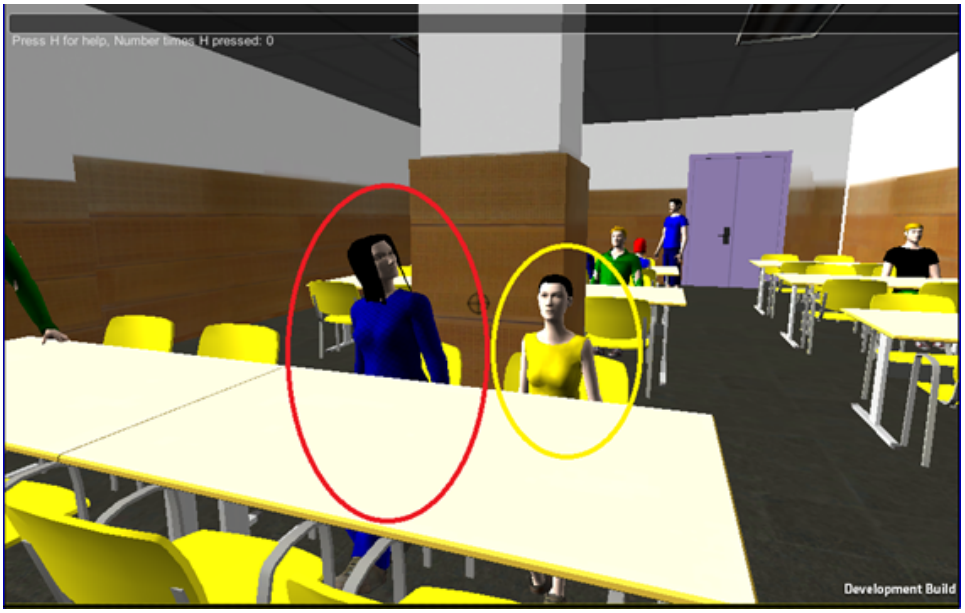

Figure 4.7: Survey 1 scene 3 described person

4.5.4 Scene 4

As seen in the Figure 4.8 the target character is the one furthest away from the user.

The data obtained from the experiments was the following:

- **Used Algorithm:** Incremental
- **Shown Description:** The boy in the red t-shirt who is sitting down with spike blonde hair. He is far.
- **Number of Correct Clicks:** 13/27
- **Mean H Pressed:** 0,33 times
- **Mean Time:** 3,50 seconds

This scene is one of the lowest scoring scenes. Given the time and the number of times the H key was pressed, it seems that the problem was that the description was too complex to understand by reading it just one time.

One of the key characteristic of this character is his distance from the user. A possible solution would be to specify first the position of the character and then use the others descriptors. This way the user would have a clear idea of where the target is in the scene. In Figure 4.9 we have a closer look of the target person.

Figure 4.8: Survey 1 scene 4 starting position

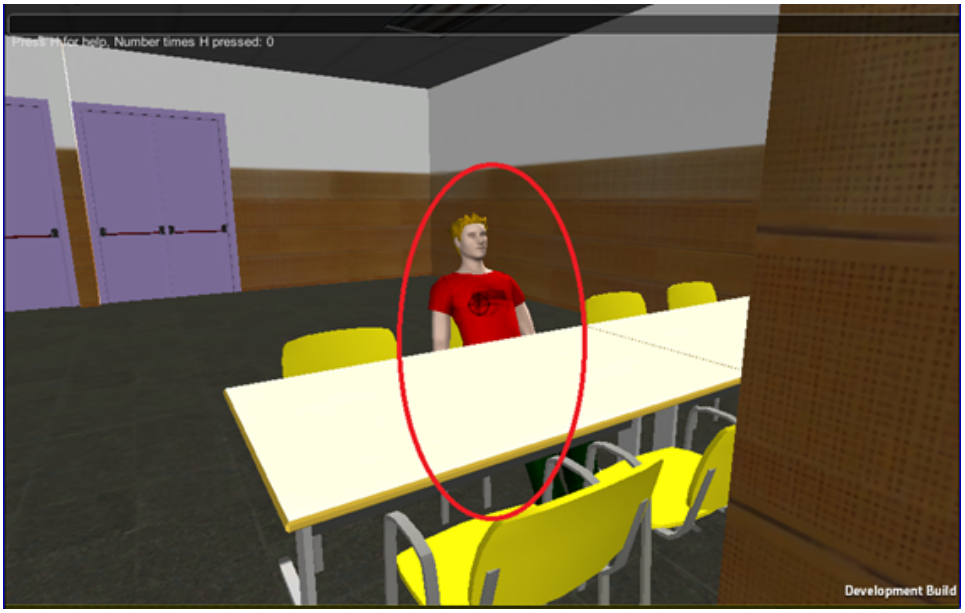

Figure 4.9: Survey 1 scene 4 described person

4.5.5 Scene 5

This scene has one of the highest mean times. This means that, in average, all the users spend more time in this scene than in the others.

The data obtained from the experiments was the following:

- **Used Algorithm:** Greedy
- **Shown Description:** The girl in the red shirt with redhead hair.
- **Number of Correct Clicks:** 17/27
- **Mean H Pressed:** 0,22 times
- **Mean Time:** 7,40 seconds

We believed that this was caused because the description did not have a definition of where, in the space, the target character was, and this made the need for the user to wander around the scene in search for the target. As a result, we believed the medium number of correct clicks, 17 out of 27, was produced by frustration from users that could not find the target. The target can be seen from afar in Figure 4.10

In addition, near the target, as can be seen in Figure 4.11, there was a character with the same color characteristic but with a blouse instead of a shirt; some users were confused by that and choose the wrong character. We believed this can be avoided by discerning between pieces of clothing that look alike and ones that do not, making the description fit this criteria.

Figure 4.10: Survey 1 scene 5 starting position

Figure 4.11: Survey 1 scene 5 described person

4.5.6 Scene 6

The objective of this scene was to test how users responded when the target was next to a person with a very distinctive posture. Even when the target was far, as can be seen in Figure 4.12.

The data obtained from the experiments was the following:

- **Used Algorithm:** NearbyPeopleDescriptionWithIncremental
- **Shown Description:** The girl in the green tank top who is standing up. She is near, next to the the girl standing pointing at something.
- **Number of Correct Clicks:** 23/27
- **Mean H Pressed:** 0,15 times
- **Mean Time:** 2,12 seconds

This was one of the highest scoring scenes: 23 out of 27. We believed this was due to the fact that the description uses a person that has a peculiar posture that none of the others characters have, as can be seen in Figure 4.13. Even if it is not the target person, using this character with a peculiar posture helps a great deal when finding the target.

It is also worth mentioning that this scene has a very low average time, meaning that users quickly found the described person.

Figure 4.12: Survey 1 scene 6 starting position

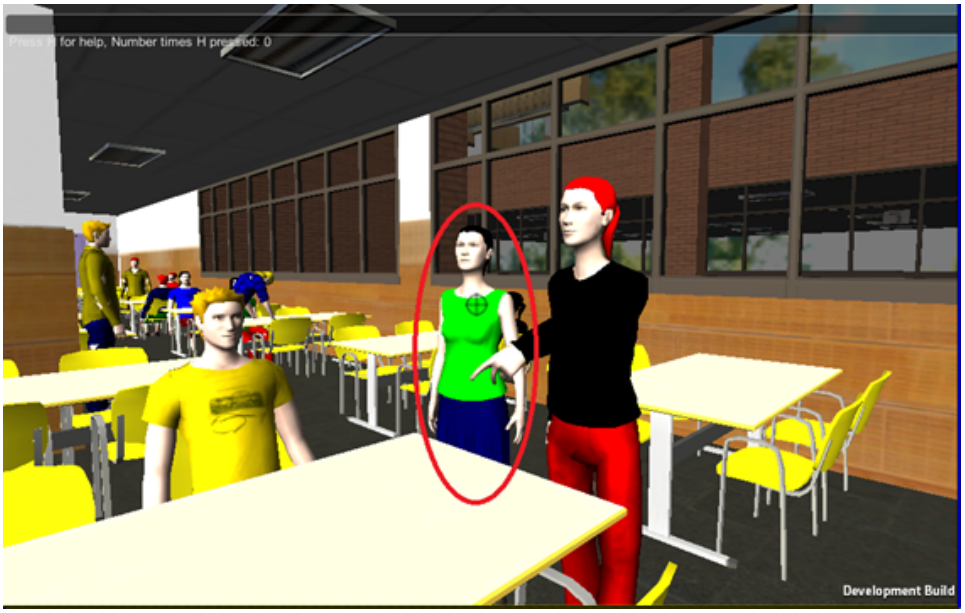

Figure 4.13: Survey 1 scene 6 described person

4.5.7 Scene 7

This scene was used as a control question; the description is very similar to the first scene and the person is in a very similar location, as can be seen in Figure 4.14.

The data obtained from the experiments was the following:

- **Used Algorithm:** NearbyObjectsWithGreedy
- **Shown Description:** The boy in the red rolled up sleeves shirt near the window.
- **Number of Correct Clicks:** 24/27
- **Mean H Pressed:** 0,04 times
- **Mean Time:** 2,99 seconds

The correct clicks were almost the same as in scene 1. The real difference is in the mean time that has decreased from 3.79 in scene 1 to 2.99 in this scene 7. We believed that at this point the user has come accustomed to the controls and can find people more quickly. This means that it is clear that we have to take into account the user ability to move when weighting the algorithm's efficiency. In Figure 4.15 we have a closer look of the target person.

Figure 4.14: Survey 1 scene 7 starting position

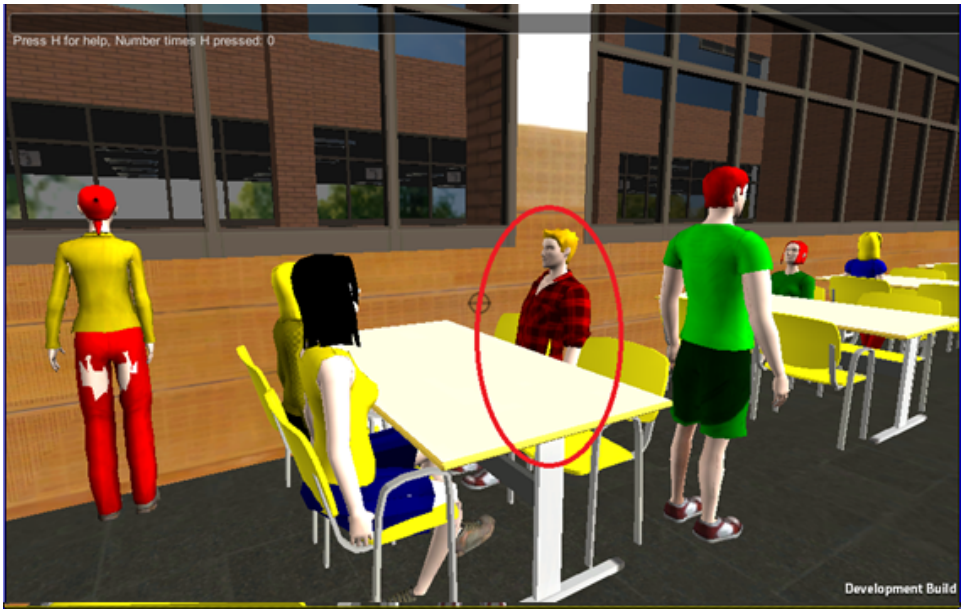

Figure 4.15: Survey 1 scene 7 described person

4.5.8 Scene 8

Less than half the users found the correct answer in this scene, and in average very little time was spent.

The data obtained from the experiments was the following:

- **Used Algorithm:** Incremental
- **Shown Description:** The boy in the blue rolled up sleeves shirt with spike redhead hair. He is near.
- **Number of Correct Clicks:** 12/27
- **Mean H Pressed:** 0,07 times
- **Mean Time:** 0,70 seconds

We believe that the low amount of correct answers was due to the fact that there was a character with a similar description next to the user's starting point, as can be seen in Figure 4.16 by the yellow circle. The only difference between this character and the target is the color of the hair. With the long description that was generated, the users gave more weight to the position of the character than to the color of his hair.

It is worth noticing that the target person has a very specific posture: leaning on his head, as can be seen in Figure 4.17. If the algorithm would have chosen to include this in the description, we believed that more people would have answered correctly. In addition, the algorithm for discerning if a character is far or near is very rudimentary, making some wrong description as in this case.

Figure 4.16: Survey 1 scene 8 starting position

Figure 4.17: Survey 1 scene 8 described person

4.5.9 Scene 9

Few people got a good result in this scene. It is worth noting that at this point, users started complaining about the length of the survey. This caused some users to lose interest and stop giving attention to details.

The data obtained from the experiments was the following:

- **Used Algorithm:** NearbyObjectsWithGreedy
- **Shown Description:** The boy in the green shirt sitting with redhead hair near the column.
- **Number of Correct Clicks:** 13/27
- **Mean H Pressed:** 0,30 times
- **Mean Time:** 1,66 seconds

We believed that less than half the users were correct in this scene because of the character, which can be seen in Firgue 4.18 in a (yellow circle). He has very similar features to the target character, except for the color of the hair. In Figure 4.19 we have a closer look of the target person.

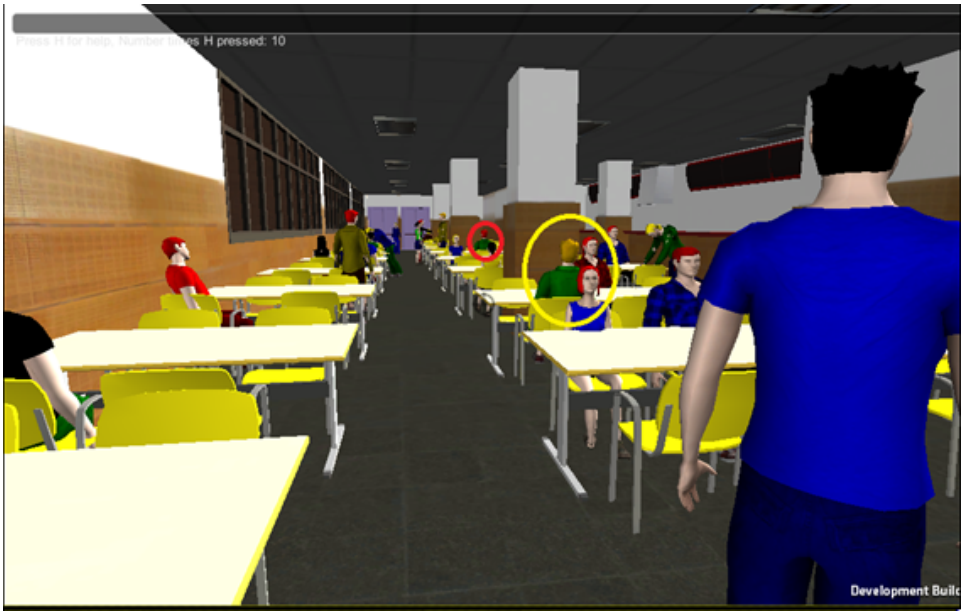

Figure 4.18: Survey 1 scene 9 starting position

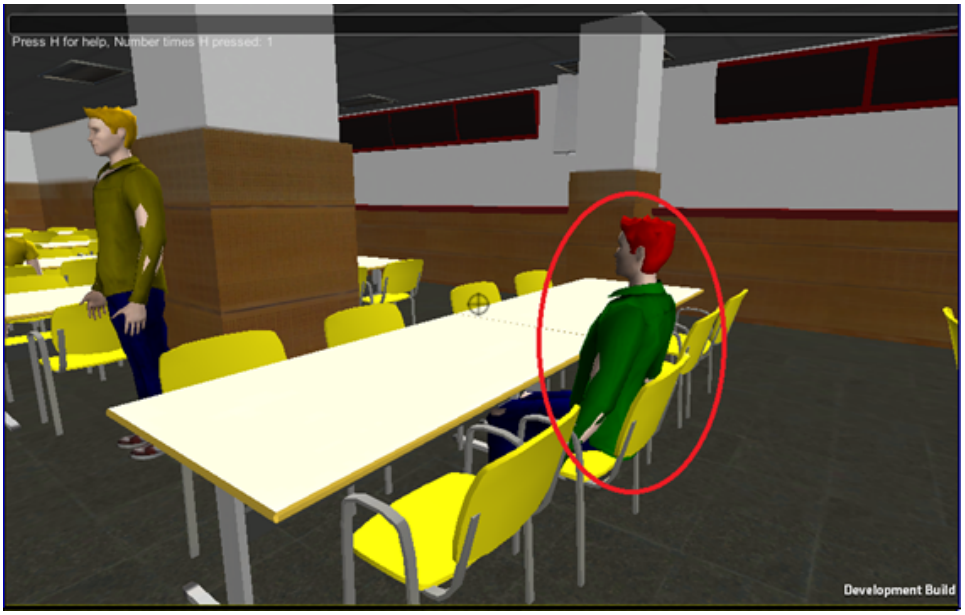

Figure 4.19: Survey 1 scene 9 described person

4.5.10 Scene 10

This final scene was done with the intention of giving the user an easy person to find.

The data obtained from the experiments was the following:

- **Used Algorithm:** Greedy
- **Shown Description:** The girl in the blue shirt standing, leaning on a table.
- **Number of Correct Clicks:** 27/27
- **Mean H Pressed:** 0,07 times
- **Mean Time:** 0,82 seconds

All users got a correct answer in this scene. The starting position can be seen in Figure 4.20. It is worth noting that the description is very small and does not give much detail. We think that the success of this scene was due to the fact that the description uses the character peculiar posture to describe her as can be seen in Figure 4.21. Using this posture and adding just enough to be able to distinguish her from the girl besides her, was enough to make all users get a correct click.

Figure 4.20: Survey 1 scene 10 starting position

Figure 4.21: Survey 1 scene 10 described person

4.6 Conclusions

As can be seen in Figure 4.22, the incremental algorithm has the lower number of correct answers. We think that this is because the incremental does not work well when there are too many people that look like the target person and because the incremental algorithm does not use nearby objects or people.

The incremental strong point is the use of the distance, between the starting position and the target character, as a descriptor. But when the user is able to move around the scene, this description becomes invalid because the distance changes. This is why we believe that changing the description when the user has moved is a crucial point that will improve the algorithm overall efficiency.

In Figure 4.23 we can see that most correct clicks were achieved when the descriptions used nearby objects or people to describe the target, and also when the description used the peculiar posture of the target to describe it. As an example of the latest, in scene 8, the target character was resting his head on his hand: Figure 4.17.

If the algorithm would have used this feature instead of specifying him by hair color and distance, we believe that this scene would have had a much higher score. By giving a higher priority to the way a character is posing, we believe that he/she would be much easier to find.

An important aspect of the description is the way it is written. We believe that by changing the order of descriptors we can achieve a higher score. This order would prioritize position and posture over pieces of clothing.

Many users got wrong a fairly easy to find person because there was a character that look like him/her in the users field of view at the start. For example, in scene 9 (Figure 4.18), the target is fairly easy to find, but because there is a character before him and much closer in the user field of view, the user rush to click the wrong person. A way to fix this can be by specifying in the description if the target is in the user's field of view or not, and if it is, if it is far or close.

Apart from the results extracted from the survey, by being present in some of the surveys done by the users, we could gather user interface information to improve the survey. One of the complaints from the users was that the mouse sensitivity was too high. This means that the camera moved too fast when moving the mouse. Other issue that arose was that the description text was too small and users had a hard time reading it.

Algorithm	Correct clicks	Scene
Greedy	26	10
NearbyObjectsWithGreedy	24	7
NearbyPeopleDescriptionWithIncremental	24	3
NearbyPeopleDescriptionWithIncremental	23	6
NearbyObjectsWithGreedy	23	1
NearbyObjectsWithGreedy	18	2
Greedy	17	5
NearbyObjectsWithGreedy	13	9
Incremental	13	4
Incremental	12	8

Figure 4.22: Survey 1 numbers of successes

Correct clicks	Description
26	The girl in the blue shirt standing, leaning on a table
24	The boy in the red rolled up sleeves shirt near the window.
24	The girl in the blue sweater with black hair. She is near. Next to the the girl in the yellow tank top
23	The girl in the green tank top who is standing up. She is near. Next to the the girl standing pointing at something
23	The boy in the black shirt sitting near the window.
18	The boy in the blue t-shirt sitting with black hair near the window.
17	The girl in the red shirt with redhead hair
13	The boy in the green shirt sitting with redhead hair near the column.
13	The boy in the red t-shirt who is sitting down with spike blonde hair. He is far.
12	The boy in the blue rolled up sleeves shirt with spike redhead hair. He is near.

Figure 4.23: Survey 1 descriptions sorted by correct clicks

Chapter 5

Perspective-Based Generation of Descriptions

"Every puzzle has an answer."

Professor Layton (Professor Layton and the Curious Village)

5.1 Changes According to Previous Results

We received feedback from some users (after the analysis of the first survey) that completed the previous survey. This information was received when a user was completing the survey and we were standing near them during the process. We asked them what they thought of the survey and they answered our questions.

5.1.1 Detected Problems

The following lines determine the information obtained from that feedback.

- **Descriptions appear for a very short time**. Users complained about the time the description appeared when the H key was pressed. They complained that they could not read the description in the *Curtain Scene*. The *Curtain Scene* is the scene that appears every time before the first person simulation starts. It has the generated description on the top left part of the screen. We can find a start button near a text (that reminds that the description is on the top left part of the screen) on the center of the screen in the *Curtain Scene*.
- **Font size is very small** and cannot be read properly. This could be related to the matter that they did not read the first generated description during the *Curtain Scene*.

- **Instructions are too long**. Users admitted that they did not read the entire instructions text. This could mean that most part of that text is unnecessary.

- **Mouse sensitivity is really high**. Users that have played *First Person Shooter* games and users that had not played, complained about this issue.

- If the user moves the mouse widely, the web **application loses the focus on the simulation**. This could bring problems when clicking a person in the simulation. The user thinks that is clicking a person but he is clicking an element of the web page, not an element in the simulation. This could make the user thinks that the application is not working properly.

5.1.2 Solving Issues

The first approach for the new survey is to solve the minor issues explained before. We have made the following changes:

- **The H key implementation** is eliminated. People did not have time to read properly the descriptions. Moreover, our intention is to generate descriptions dynamically. That means that descriptions change over time. We determined that the H key process is incompatible with the new realtime descriptions generation. So, the H key element is discarded.

- The font size is changed from 12 to 19. It is almost an increase of a 60% of the font size. We readjusted the GUI in order to have it beautifully displayed.

- The **instructions** are reduced considerably. We determined that the basic important information that the user must know during the simulation is the one shown in Figure 5.1.

- **Mouse sensitivity** has changed from 0.1 to 0.05. That means that the sensitivity of the mouse is half the previous sensitivity. That should make the user have a better experience.

- **The focus problem** is solved. Adding simple instructions to the scenes, we manage to lock the mouse on the screen when the application deployed on a Web Browser.

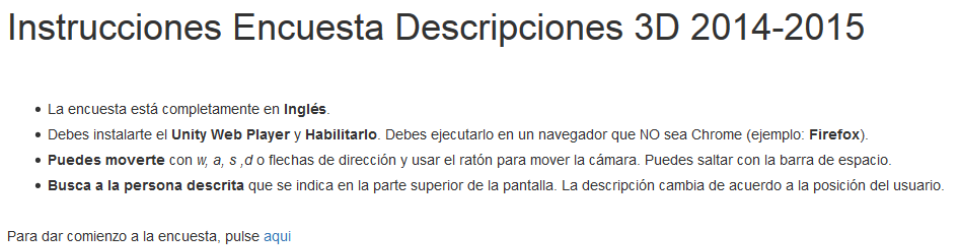

Figure 5.1: New reduced instructions

5.2 Perspective-Meta-Algorithm

The next step is to decide how our *Perspective-Meta-Algorithm* must works. This new algorithm is based on the previous one but it adds some perspective relative annotations to the currently descriptions. Our new Perspective-Meta-Algorithm generates descriptions dynamically based upon two main components:

- **Attributive or Environment Descriptions**. These descriptions refer to the static characteristics of an individual and its environment. This characteristics are elements that cannot change during the simulation. They can refer for example to the color of the shirt of a person or his postion relative to the window (this position cannot change during the simulation). A description generated with this component is the following: *The boy in the black shirt near the window*. The previous Meta-Algorithm is the responsable of generating these descriptions.
- **Perspective or User Relative Descriptions**. These descriptions have to be generated in real time according to the situation of the user relative to the situation of the described person. This adds the perspective or relative elements to the previous Meta-Algorithm. There are serveral relative elements involving the user and the described person. We have to decide which elements can make our descriptions better.

5.2.1 Perspective-Based or User-Relative Descriptions

This is the relative information we decided to obtain in real time so we could improve the descriptions of our algorithm:

- **Distance from the User to the Described Person**. The previous meta-algorithm already took this information into consideration. However, it was prepared only for static situations (scenes where you have only one point of view and you cannot move through the scene).

Nevertheless, this distance has to be calculated for each frame (real time) and the new algorithm uses it to generate a new description. We have decided to divide the space into three areas because it is necessary to define proximity zones in order to generate good descriptions: near, not so far and far. The algorithm stores this information and decides if it has to be shown to the user in order to find the described person.

- **Visibility from the User to the Described Person**. We thought that it could be useful to show the user the visibility status of the described person from his point of view. The user is informed in real time if he is seeing the described person or if the described person is behind a column or behind a person from his position. This information can be really useful in situations where the user is near. Knowing if you are seeing the described person in that situation can make the described person easier to find.

- **Distance from Certain Points in the Scene to the Described Person**. We have decided to put certain control points in our scenes. These points gather information about characters near them. For example, if we put a point at the back of the canteen and a described person is near that point, the algorithm stores the information that the character we seek is near it. However, the position *"back of the canteen"* is relative to the position of the user. If the user is standing near the point's location, the *"back of the canteen"* is the one that the user finds at the other side of the canteen, as seen in Figure 5.2.

  On the other hand, if the user is standing far from that point, exactly at the other side of the canteen, the user will perceive the *"back of the canteen"* the location that is near the control point as seen in Figure 5.3. This means that the "back of the canteen point" is relative to the user's location. Our algorithm must bear these aspects when generating the description.

5.2.2 Detection Script

In order to generate the new perspective or user relative descriptions is necessary to retrieve the corresponding information. We have decided to create a new script called *Detected* that obtains and manages that information. That script is attached as a *Component* to the described person in runtime. This script obtains the necessary information to generate the descriptions explained in Section 5.2.1.

It stores the needed flags (for example: isVisible) and the textual description data. We are going to explain which Unity's elements are used in order to calculate that new flags.

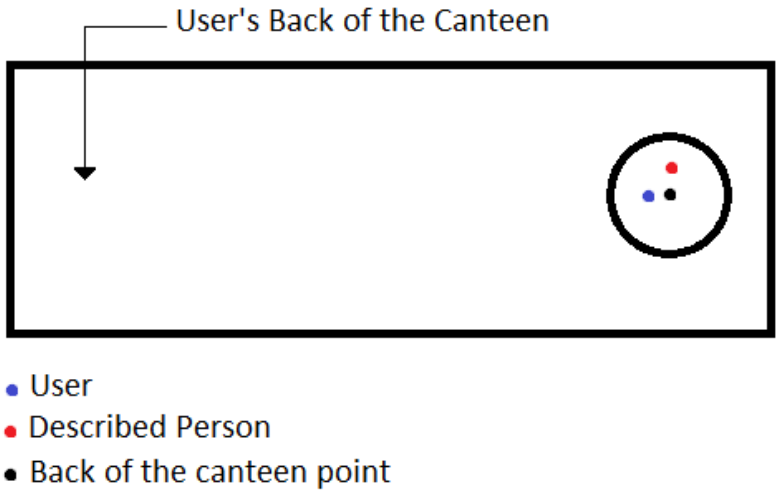

Figure 5.2: Back of the canteen when the user is near the point

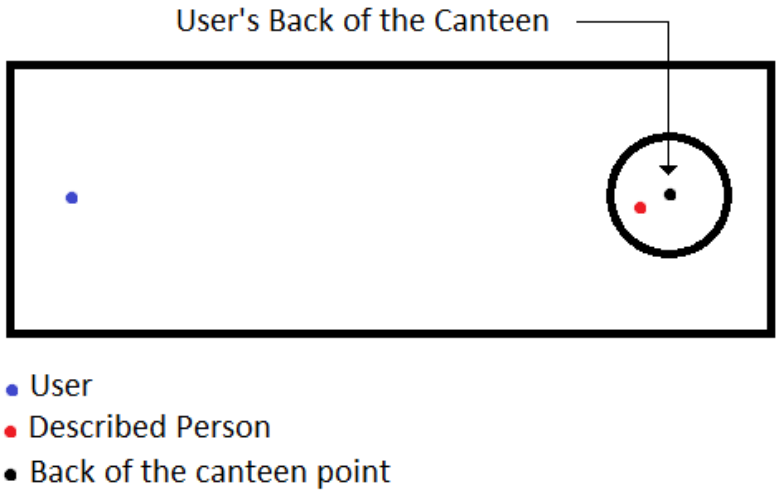

Figure 5.3: Back of the canteen when the user is at the other side of the canteen

5.2.2.1 Calculating Distace from User to the Described Person

Position in Unity is represented as a *Vector3* just as we have explained in Section 2.3.3. The user and the described person have a Vector3 in the *Transform Component* which represents the position. In order to calculate the euclidean distance between two points, we can use the static function

Distance defined in the *Vector3* class. Unity provides this function.

The distance calculation is done in the *Update* method (see Section

The longest distance in the canteen is about 15 units. We decided to divide the space in order to generate good descriptions. The cuantification consists in:

- if the distance is more than 10 units, we consider the described person to be *far* from the user.
- if the distance is between 3 and 10 units, the described person is *not so far*.
- If the described person is less than 3 units from the user, we consider it to be *near* the user.

5.2.2.2 Visibility from the User to the Described Person

Visibility between the user and the described person is really important in order to locate the described person properly. The implementation is based on two elements:

- **Camera Events:** We consider the camera area the spatial area that is inside the camera view. There are two interesting events that are generated when a *GameObject* with a *Render Component* enters or abandons the camera area: *OnBecameVisible()* and *OnBecameInvisible()*. *OnBecameVisible()* is the event that is generated when the *GameObject* enters the camera area and *OnBecameInvisible()* is generated when the *GameObject* abandons the camera area. We have defined those methods and a variable called *isOnCameraArea* in the *Detected* script in order to obtain in realtime whether the described person is in the camera area or not.
- **Raycasting:** As explained in Section 2.3.4, this element is used to detect collisions from a ray shot from one point with one direction. We have decided to shoot a ray from the eyes of the user to the head of the described person (we call it *Ray 1*). That is necessary to detect if the user is seeing the described person directly or there is an object is between them. We can shoot a ray from one point to another by calculating the direction of the ray we want to shoot. That direction can be easily calculated with the following mathematical operation: *direction = originPosition - destinationPosition*. The objects that can be in front of the user are: columns or other characters. The generated description depends on whether the described person is visible from the user's point of view or there are obstacles between them. This code must be in the *Update* method because this can change due to the user's movements.

By mixing the *Camera Events* and *Raycasting* we can obtain objects detection when the described person is on the camera area.

In addition, the algorithm must detect whether the described person is to the right or left from the user. We want to detect if the described person is behind the user too. This detection should be activated only when the described person is not on the camera area. The implementation of the solution to this problem can be a little complex

Another ray (*Ray 2*) is shot from the middle of the screen perpendicular to the user's screen. The idea is to calculate the angle that form this ray and the ray which detects objects between the user and the described person (we call it *Alpha Angle*). The described person's relative position to the user depends on the value of the *Alpha Angle*. We use *Vector3.Angle* to obtain the angle that two directions describe (not Rays).

That method returns the angle value without sign, but the angle is useless without its sign. However, we can obtain the sign of the angle using the method *Vector3.Cross* (which receives the directions which make the angle as paratemers). That operation implements the mathematical cross product between two vectors (in our case two directions), and returns another vector. If the value of the returned vector is negative, that means the angle is negative too.

Depending on the value of the *Alpha Angle*, the location of the described person relative to the user changes:

- if the **angle is positive and less than 130**, the described person is to the **user's right**.
- if the **angle is negative and more than -130**, the described person is to the **user's left**.
- if the **angle is between 130 and -130**, the described person is **behind the user.**

We can see a simplified representation of these mechanics in Figure 5.4. That figure represents the case when the described person is in the camera area and behind the column. These descriptions are shown only when the described person is not on the camera area. We chose the zone between -130 and 130 degrees to be the *user's back* zone because it feels comfortable and realistic during the simulations.

5.2.2.3 Distance from Certain Points in the Scene to the Described Person

Distance calculation from the user to the points on the scene works exactly the same way as explained in Section 5.2.2.2. Each point has a *Transform*

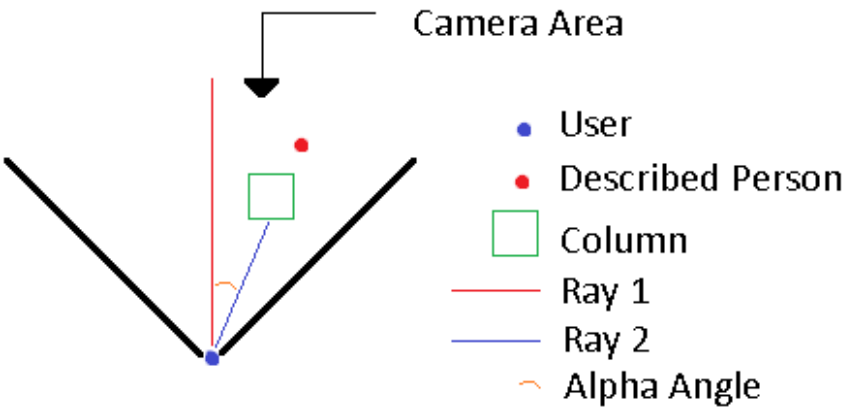

Figure 5.4: Visibility Detection Diagram

component. That means that we can use the method *Vector3.Distance* too. This code must be written inside the *Update* method too.

5.2.3 Perspective-Meta-Algorithm Workflow

This new algorithm inherites part of its workflow from the Meta-Algorithm as shown in Figure 5.5. It is an extension of the previous one and adds several properties that are necessary to generate perspective based descriptions. It adds a dependency to the *Detection* script (see Section 5.2.2). This is necessary because that element is the one that stores and updates the information about the relative information necessary to generate the new descriptions. That information is recalculated each frame and communicated inmediatly to the Perspective-Meta-Algorithm. Then, the GUI is updated to show the new description.

Each description can be composed of sub-descriptions. A sub-description is a description that contains partial information of a full description. Every sub-description is generated according to the retrieved data that the *Detected* script obtains. Each descriptions can contain four possible sub-descriptions:

- **Meta Description:** The description generated by the original Meta-Algorithm. User's positional relative information must be eliminated from this descriptions because the new Perspective-Meta-Algorithm is the main responsible to obtain that information. For example, in some descriptions the algorithm returns that the described person is near you. That part has to be erased.

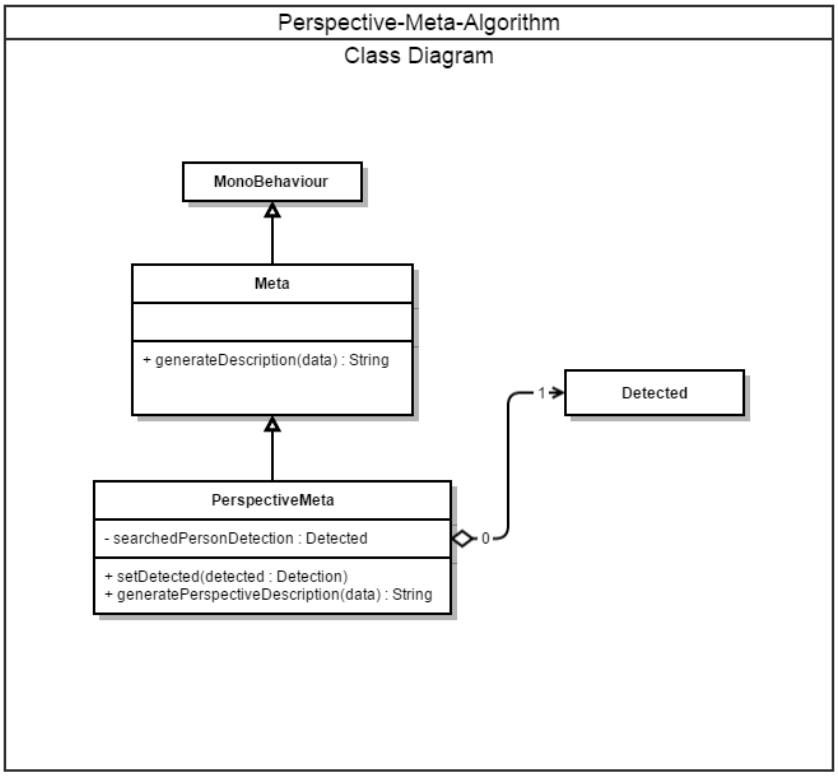

Figure 5.5: Simple Class Diagram of the Perspective-Meta-Algorithm

- **Description of the Static Points:** This description contains the information of the static points scattered all over the scene. Only one point of this kind is on the scene. It is the one at the back of the canteen as explained beforehand.
- **Description of the Visibility:** It contains the information about the visibily of the described person from the user's point of view. For example, that the described person is behind a column.
- **Distance between the Described Person and the User:** It contains a textual explanation of the distance between the described person and the user.

The last three sub-descriptions are generated and stored inside the *Detected* script. The algorithm checks certain flags of the *Detected* script sequentially each frame and the information needed for the description is generated according to those flags. The flag checking flow is the one determined next:

- First it checks the distance between the user and the static points that we have previously put into the scene. In our scenes, a point has been put at the back of the canteen, keeping in mind the explanation of what we consider back of the canteen explained before. If the distance to the user is more than a constant that we have previously defined (in our case, we consider the user is far from the point when the distance between the user and the point is greater than 10 just like we explained in Section 5.2.2.1) and the described person is near that point, the generated description contains information about the proximity of the described person to that point.

- If none of the conditions is met, the algorithm checks if the described user is on the camera area. If it is not within the camera area, the generated description must contain the positional references of the described person to the user. That means that it indicates if the described person is to the left or right direction of the user. It shows the user if the described person is behind too. That should make the user move the camera to the direction the description indicates.

- If the described person is within the camera area, it checks the absolute distance between the described person and the user. It indicates the user whether he is near, not so far or far from the described person.

By mixing sub-descriptions and flag checking, the new perspective descriptions are generated. Not every single sub-description is shown to the user every time. The order of appearance of sub-descriptions not always is the same as well. As a result, in each case a specific order and sub-description appear in the generated description.

Chapter 6

Final Evaluation

"Prepare for unforseen consequences."
The G-man (Half life 2)

6.1 Purpose of the Second Survey

After the first survey, we decided to implement a system that generates descriptions in real time according to the user's positional tracking. This is the last objective mentioned in Section 1.2:

Therefore, this second survey has two main objectives:

- To test the solutions to the problems that were generated during the implementation of the first person simulation mechanics.
- To test the dynamic generation of descriptions.

6.2 Dynamic Generation of Descriptions

The descriptions should be generated dynamically during the described person search (first person scene) and during the curtain scene. The previous curtain scene was inefficient and unmaintainable (some code was duplicated). The corresponding scene characters were loaded during the curtain scene, and the described character was selected before the characters were loaded. All the data of the characters of the scene was destroyed when the user clicked the start button. Then, the data was reloaded again and the first person scene started. Two scenes were involved in this process, the curtain scene and the first person scene. That means that two times the character data was loaded: one during the curtain scene and another time during the first person scene start.

It was necessary to rebuild the curtain scene and the scene's loading process from scratch because the data could not be destroyed after the first

description is generated. If the character's data was destroyed when the description is generated dynamically, the whole application collapsed.

We have decided to load one scene with the whole set of data just once for each described character. That unique scene must contain the data of the characters, the model of the canteen and the first person controller. The first person controller must be deactivated when the scene is just loaded because if it is activated, you could move over the scene during the new curtain scene. The curtain scene is no longer a Unity scene, but a canvas that is painted over the first person scene. The first person controller is activated and the necessary curtain UI elements disappear when the user clicks the start button.

The new UI curtain elements cannot be implemented easily as in the previous UI implementation. The curtain UI implementation must be changed completely. We decided to change the UI elements of the whole application in order to maintain coherence between the new curtain UI elements and the other UI elements (which includes ages, gender and FP selection scenes).

We bear in mind the feedback the users gave us when we deployed the first survey during this process. Eficiency is improved noticeably because the data is loaded just for each described character (not twice like the previous implementation worked). UI elements are optimized for Web deployment and resolution (960 x 600). We can see the new UI curtain canvas in Figure 6.1.

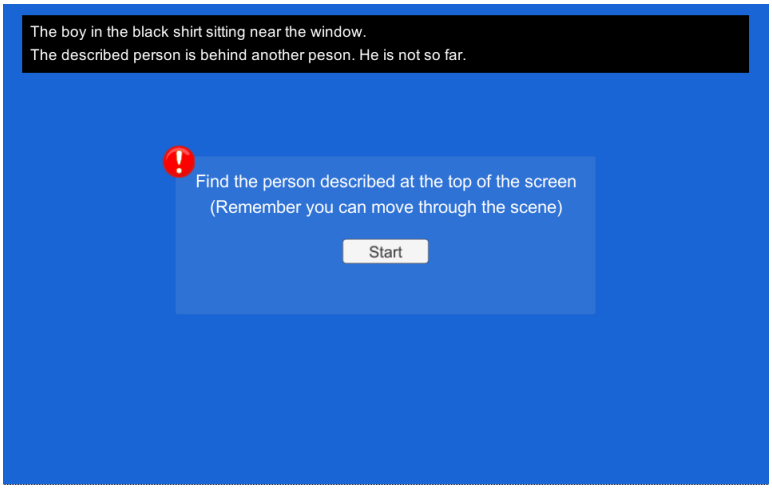

Figure 6.1: New Curtain UI Canvas

6.3 Deployment

The survey is deployed on a server of our University (overriding previous dataf).
The link is: *http://tot.fdi.ucm.es/descripciones/*

6.4 Second Survey Result Analysis

Because the main focus of this survey was to test the improvements of the algorithm by comparing the results from this survey to the first survey, the described characters the user had to find are the same in both surveys. This way, a reliable comparison can be made and the result would be directly influenced by the algorithm's changes.

Following this mindset, the amount of people that completed the survey was twenty seven, the same as the first survey. Most of them were in their twenties and a few in their thirties, 37% were female. Additionally, people that had completed the first survey were asked how they felt about the changes and if it made the survey easier.

We present the individual results for each scene. Each scene was selected specifically to test the algorithm performance when the user is able to move and when the described person is not visible. In each individual result the first picture corresponds to how the user starts in the scene without any movement, and the second is where the described person is.

To clarify who the person is, a (red) circle has been drawn around the described person. Among the statistics that the Survey Analyzer provides, Mean Time is the average time all users took to complete a certain scene.

6.4.1 Scene 1

As can been seen in Figure 6.2, the described person is behind a character and he is far, so the user has to move inside the scene. This way the user can get accustomed to the controls and realize that is mandatory to move to find certain people. Once the user moved, it was not difficult to find the described person. In Figure 6.3 we can see the target from a closer point of view.

The data obtained from the experiments was the following:

- **Used Algorithm:** PerspectiveNearbyObjectsWithGreedy
- **First Shown Description:** The boy in the black shirt sitting near the window. The described person is behind another person. He is not far.
- **Last Shown Description:** He is near. You can see the described person.The boy in the black shirt sitting near the window.
- **Number of Correct Clicks:** 24 / 27
- **Mean Time:** 5,0 seconds

This scene was key for users that were not accustomed to this type of controls. The users with higher times in this scene are the ones that were not used to the controls. In despite of this, this scene had a high number of correct clicks: 24 out of 27 people found the target person.

Here we can see how the new algorithm differs from the previous one by detailing at the start that the described person is behind another person. In the previous survey (Section 4.5.1), 23 out of 27 found the correct person.

Figure 6.2: Survey 2 scene 1 starting position

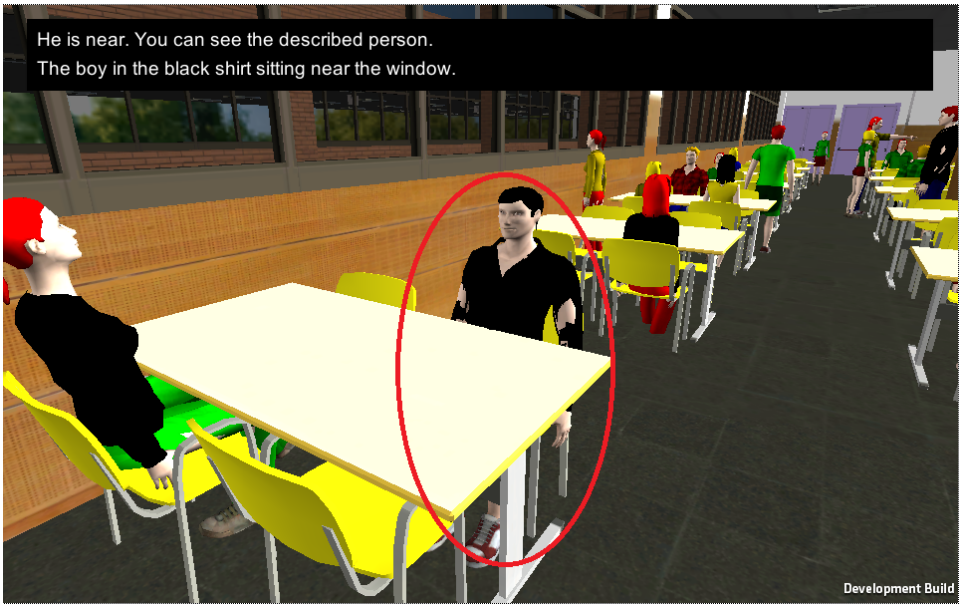

Figure 6.3: Survey 2 scene 1 described person

6.4.2 Scene 2

In Figure 6.4 we can see that the target person is in the field of view of the user. The main purpose of this scene was to test the algorithm performance when having many characters that look like the target person and are close to the target. As it can be seen in Figure 6.5, the target character is next to many people sitting, with blue shirt and with black hair. This caused many people to get confused and choose the wrong person.

This was one of the lowest scoring scene in the first survey (Section 4.5.2), with 18 out of 27 people getting the correct answers.

The data obtained from the experiments was the following:

- **Used Algorithm:** PerspectiveNearbyObjectsWithGreedy
- **First Shown Description:** The described person is far from you. The boy in the blue t-shirt sitting with black hair near the window.
- **Last Shown Description:** He is near. You can see the described person.The boy in the blue t-shirt sitting with black hair near the window.
- **Number of Correct Clicks:** 24 / 27
- **Mean Time:** 5,3 seconds

Having the description begin with the distance of the user to the target person helps many users to find the target quicker and focus more on the differences between all the characters that are next to the target that look like him.

It is also worth noting that because the description changed depending on the user's distance and field of view, the users were more attentive to what the description said. This avoided the issue from the previous survey were, because the description was fairly easy, no one felt that it was necessary to read the description again, causing the users to make a rapid decision and choose the wrong person.

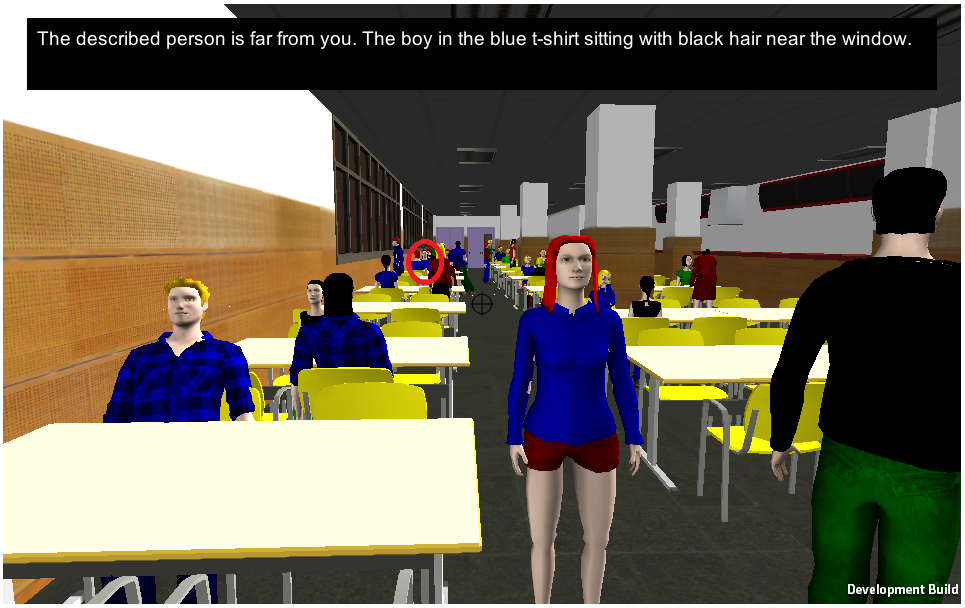

Figure 6.4: Survey 2 scene 2 starting position

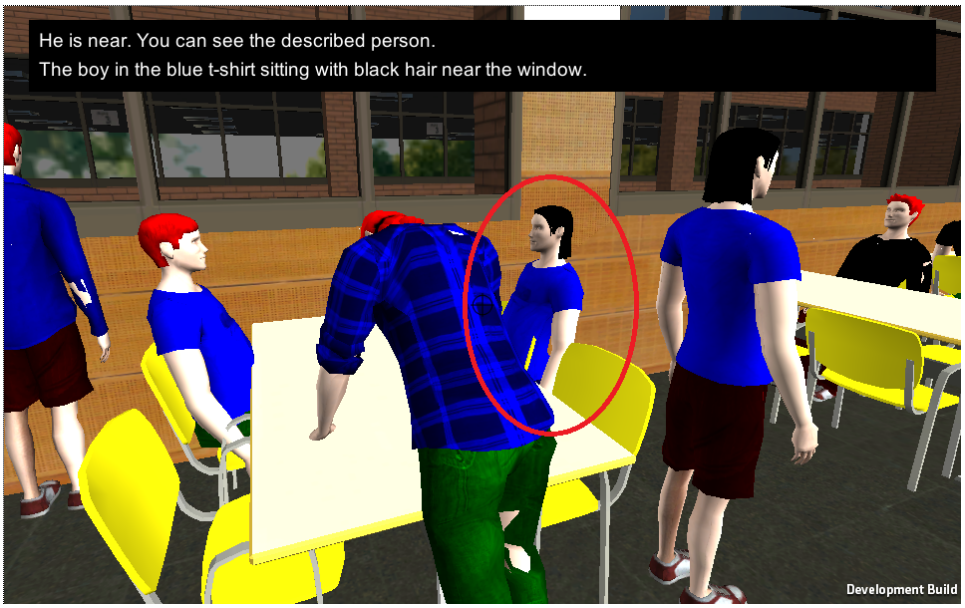

Figure 6.5: Survey 2 scene 2 described person

6.4.3 Scene 3

This is the first scene were the described person is behind an object, as can been seen in Figure 6.6. Again we can see how the new algorithm details how the person is behind a column, helping the user find the target person. As in the previous survey the meta-algorithm chose NearbyPeople algorithm, making this scene easier by describing the girl in the yellow tank top (yellow circle) that is visible from the start and is next to the target.

In Figure 6.7 we can see the target from a closer point of view and see how the new algorithm details that the person the user is looking for is near and is in their field of view.

The data obtained from the experiments was the following:

- **Used Algorithm:** PerspectiveNearbyPeopleDescriptionWithIncremental
- **First Shown Description:** The girl in the blue sweater with black hair. Next to the the girl in the yellow tank top. The described person is behind a column. She is not far.
- **Last Shown Description:** She is near. You can see the described person.The girl in the blue sweater with black hair. Next to the the girl in the yellow tank top.
- **Number of Correct Clicks:** 25 / 27
- **Mean Time:** 3,5 seconds

In the previous survey (Section 4.5.3), 24 out of 27 found the correct person. This does not defer much from the results of the previous survey but it helps to prove that these changes were an improvement and not a downgrade.

Figure 6.6: Survey 2 scene 3 starting position

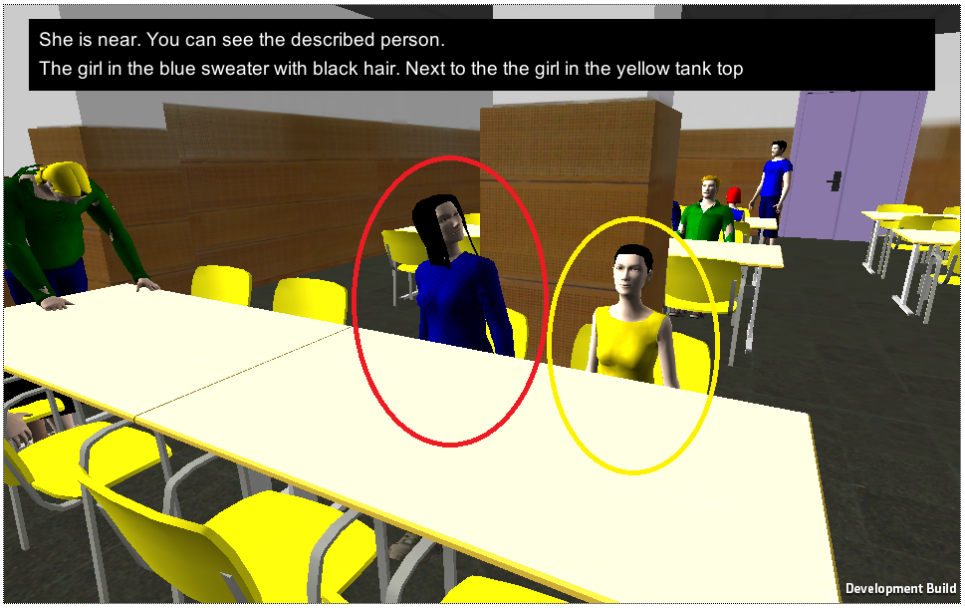

Figure 6.7: Survey 2 scene 3 described person

6.4.4 Scene 4

As can been seen in Figure 6.8, the target character is the one furthest away from the user. In the previous survey (Section 4.5.4) this scene got the lowest score, 13 out of 27.

The data obtained from the experiments was the following:

- **Used Algorithm:** PerspectiveNearbyObjectsWithGreedy
- **First Shown Description:** He is at the back of the canteen. The boy in the red t-shirt near the column.
- **Last Shown Description:** He is near. You can see the described person.The boy in the red t-shirt near the column.
- **Number of Correct Clicks:** 24 / 27
- **Mean Time:** 2,9 seconds

One of the problems in this scene in the previous survey was that the description was too complex to understand with just one read. To solve this, if the target is too far to see, the new algorithm details the distance first before giving much description of the target person, this forces the user to get closer and narrow the search. In Figure 6.9 we can see the target from a closer point of view and that the description is simpler than the one given in the first survey. This is possible because the meta-algorithm recognized that by having the target character separated from other characters and in a specific point in the environment, it does not need to specify posture or hair style as it did in the previous survey: "The boy in the red t-shirt who is sitting down with spike blonde hair".

Figure 6.8: Survey 2 scene 4 starting position

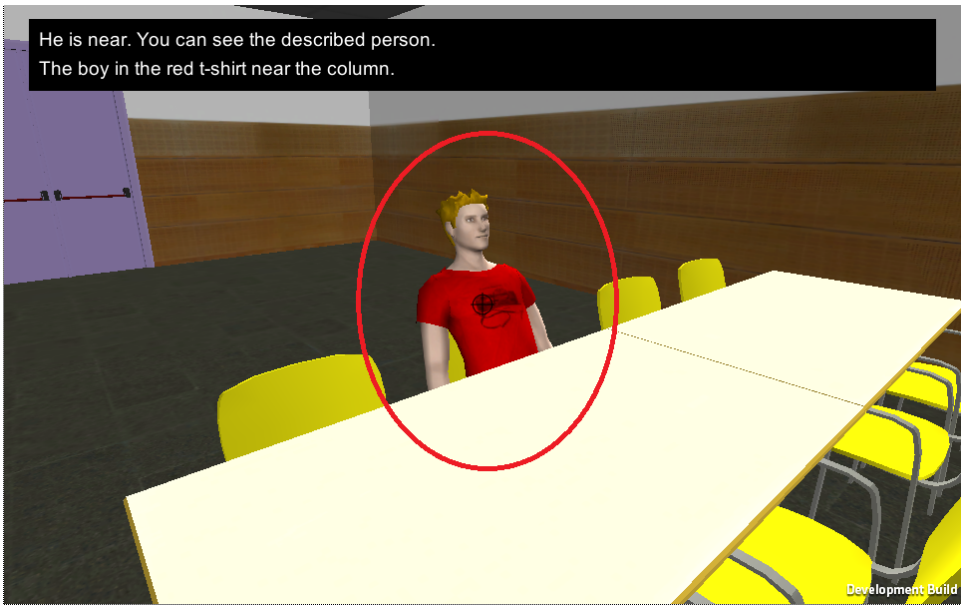

Figure 6.9: Survey 2 scene 4 described person

6.4.5 Scene 5

In the previous survey (Section 4.5.5), this scene had one of the lowest score with 17 out of 27 people having found the correct person. This was because the description did not have a definition of where, in the space, the target character was, and this made the need for the user to wander around the scene in search for the target. This caused frustration to users and made them not to find the target person.

The data obtained from the experiments was the following:

- **Used Algorithm:** PerspectiveGreedy
- **First Shown Description:** The described person is far from you.The girl in the red shirt with redhead hair.
- **Last Shown Description:** She is near. You can see the described person.The girl in the red shirt with redhead hair.
- **Number of Correct Clicks:** 21 / 27
- **Mean Time:** 4,7 seconds

As can been seen in Figure 6.10, this time the description had where the character was with respect to the user. This proved to be an improvement having more people finding the correct person. Nonetheless, not having in the description where in space the target was, was a key factor in not acquiring has many corrects clicks has other scenes in this second survey. In Figure 6.11 we can see the target from a closer point of view.

Figure 6.10: Survey 2 scene 5 starting position

Figure 6.11: Survey 2 scene 5 described person

6.4.6 Scene 6

In the previous survey (Section 4.5.6) this scene got a good score of 23 out of 27 people. This was because the target person was next to a person with a very distinctive posture.

The data obtained from the experiments was the following:

- **Used Algorithm:** PerspectiveNearbyPeopleDescriptionWithIncremental
- **First Shown Description:** The described person is far from you. The girl in the green tank top who is standing up.
- **Last Shown Description:** She is near. You can see the described person.The girl in the green tank top who is standing up. Next to the the girl standing pointing at something
- **Number of Correct Clicks:** 25 / 27
- **Mean Time:** 4,4 seconds

The description is similar to the one used in the previous survey but it is worth noting that the description first tells the user that the person is far, as can been seen in Figure 6.12, and when he/she gets close enough it changes specifying details that are easier to understand when the user is closer to the target person, as can be seen in Figure 6.13.

Figure 6.12: Survey 2 scene 6 starting position

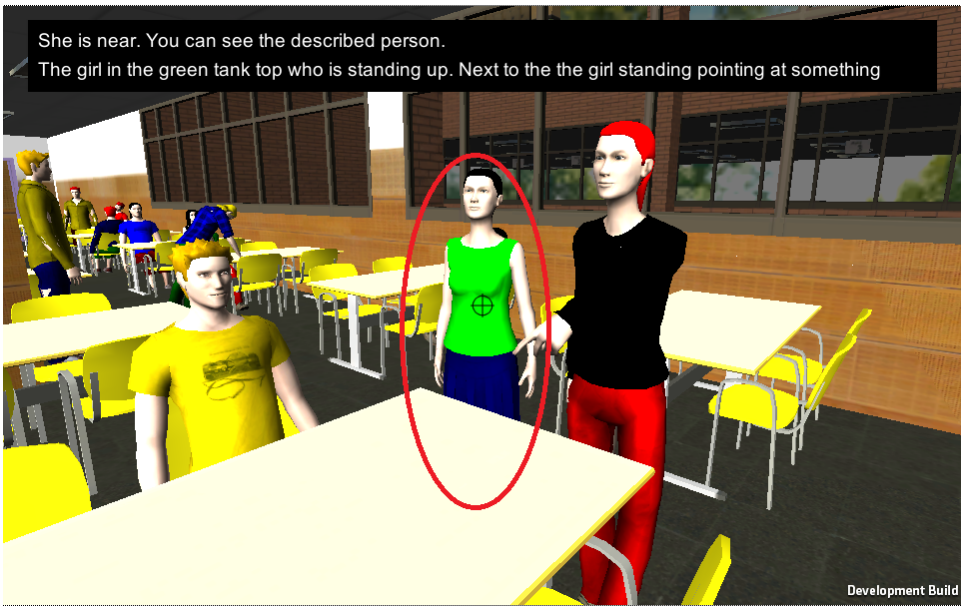

Figure 6.13: Survey 2 scene 6 described person

6.4.7 Scene 7

Like in the previous survey (Section 4.5.7) this scene was used as a control question, were the description is very similar to the first scene and the person is in a very similar location, as can be seen in 6.14

The data obtained from the experiments was the following:

- **Used Algorithm:** PerspectiveNearbyObjectsWithGreedy
- **First Shown Description:** The described person is far from you. The boy in the red rolled up sleeves shirt near the window.
- **Last Shown Description:** He is near. You can see the described person.The boy in the red rolled up sleeves shirt near the window.
- **Number of Correct Clicks:** 25 / 27
- **Mean Time:** 4,3 seconds

The correct clicks were almost the same as in scene 1. The difference is in the mean time that has decreased in this scene. We believed that at this point the user has come accustomed to the controls and can find people more quickly. In Figure 6.15 we have a closer look of the target person.

Figure 6.14: Survey 2 scene 7 starting position

Figure 6.15: Survey 2 scene 7 described person

6.4.8 Scene 8

In the previous survey (Section 4.5.8) less than half the users found the correct answer (12 out of 27), and in average very little time was spent. This was due to the fact that there was a character with a similar description next to the user's starting point, as can be seen in Figure 6.16 by the yellow circle. The only difference between this character and the target is the color of the hair as can be seen in Figure 6.17.

The data obtained from the experiments was the following:

- **Used Algorithm:** PerspectiveIncremental
- **First Shown Description:** The described person is far from you. He boy in the blue rolled up sleeves shirt with spike redhead hair.
- **Last Shown Description:** He is near. You can see the described person.The boy in the blue rolled up sleeves shirt with spike redhead hair.
- **Number of Correct Clicks:** 23 / 27
- **Mean Time:** 0,30 seconds

Because now the description starts with the distance the target is from the user, he/she can quickly discard the character in the yellow circle, making this scene have a lot more correct clicks than its counterpart in survey 1.

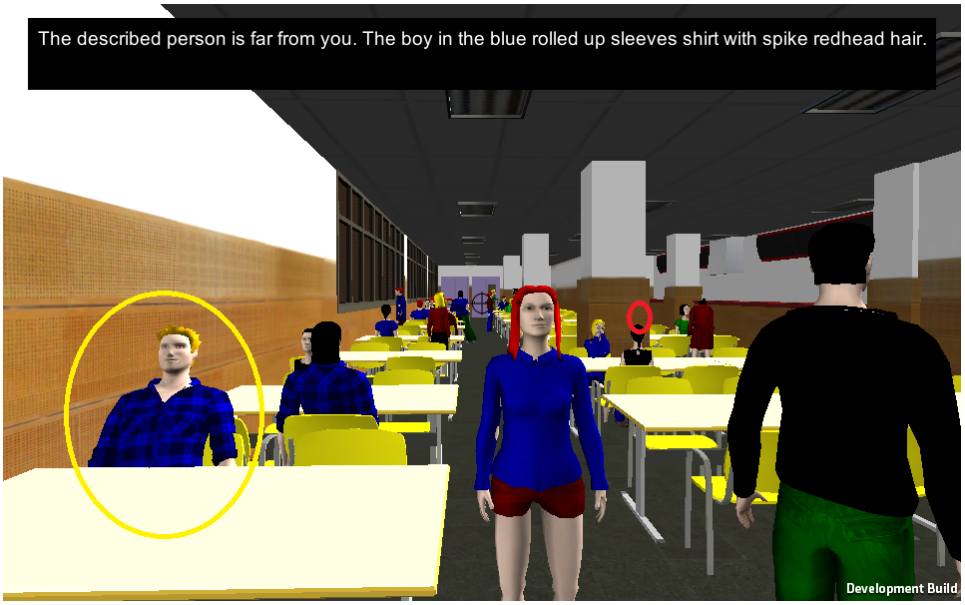

Figure 6.16: Survey 2 scene 8 starting position

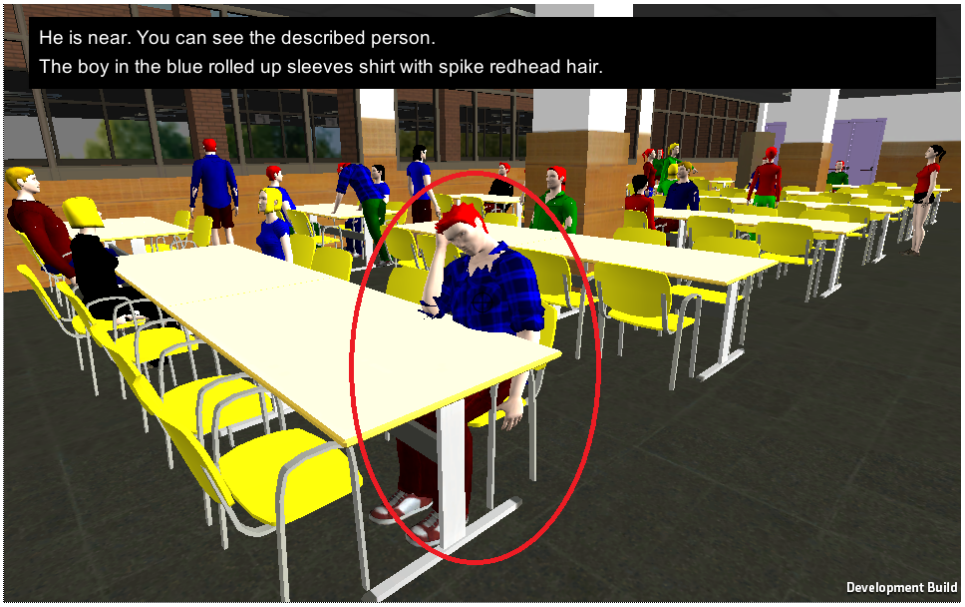

Figure 6.17: Survey 2 scene 8 described person

6.4.9 Scene 9

In the previous survey (Section 4.5.9) few people got a good result in this scene, 13 out of 27. By this point in the previous survey users started complaining about the length of the survey. This caused some users to lose interest and stop giving attention to details.

This time we did not received these complaints from the users, we believe that because the description is changing according the position and field of view of the user, making the user feel more engaged and mimicking how a real person would be giving the description.

The data obtained from the experiments was the following:

- **Used Algorithm:** PerspectiveNearbyObjectsWithGreedy
- **First Shown Description:** The described person is far from you. The boy in the green shirt sitting near the column.
- **Last Shown Description:** He is near. You can see the described person.The boy in the green shirt sitting near the column.
- **Number of Correct Clicks:** 23 / 27
- **Mean Time:** 0,48 seconds

In the previous survey, many user got a wrong answer because of the character in a yellow circle which can be seen in Figure 6.18. This time this was not an issue because the algorithm details if the person is far or not from the user. In Figure 6.19 we can see the target from a closer point of view.

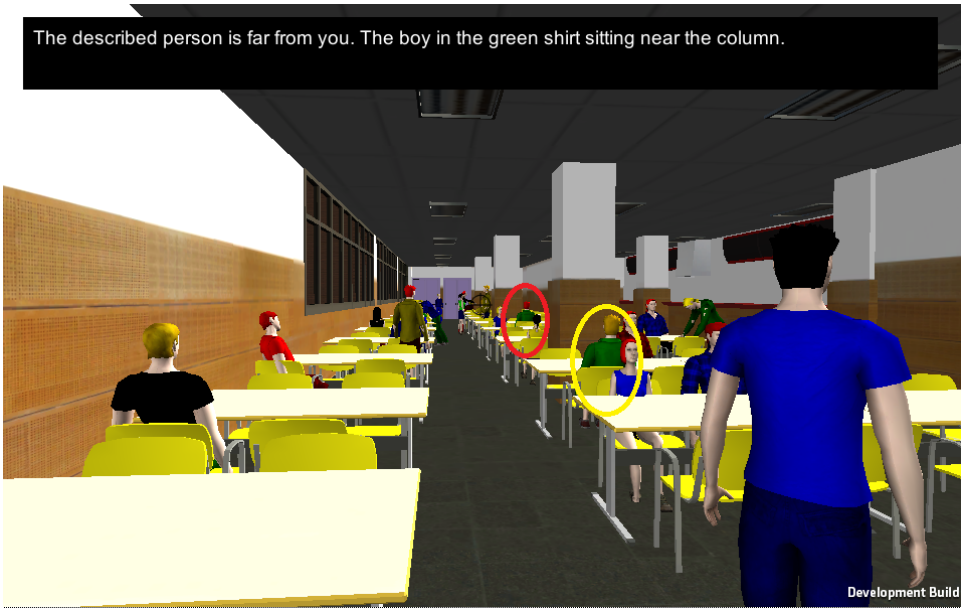

Figure 6.18: Survey 2 scene 9 starting position

Figure 6.19: Survey 2 scene 9 described person

6.4.10 Scene 10

This final scene was done with the intention of giving the user an easy person to find. As can been seen in Figure 6.20, the description starts by detailing the target person's description and not the position regarding the user. This is because the user is close enough for this information to be valuable.

The data obtained from the experiments was the following:

- **Used Algorithm:** PerspectiveGreedy
- **First Shown Description:** The girl in the blue shirt standing, leaning on a table. You can see the described person. She is not far
- **Last Shown Description:** She is near. You can see the described person.The girl in the blue shirt standing, leaning on a table
- **Number of Correct Clicks:** 26 / 27
- **Mean Time:** 0,09 seconds

In the previous survey all users got a correct answer. As stated in the analysis of this scene in the first survey (Section 4.5.10), we believed that the success of this scene was due to the fact that the description uses the character peculiar posture to describe her as can be seen in Figure 6.21.

Figure 6.20: Survey 2 scene 10 starting position

Figure 6.21: Survey 2 scene 10 described person

6.5 Conclusions

The new algorithm has a success rate of 88% (240/270). Comparing it to the previous algorithm that got 71% (194/270), we can see a major improvement in the algorithm's capabilities to adapt to different perspectives.

A lot of factors influenced the overall improvement of the algorithm. Having the algorithm detail the distance of the user to the target person and specifying if he/she was in the field of view of the user resulted in a major factor in the success of the algorithm. Also, the changes of the description in real time help the user in a more realistic way, mimicking how a real person would be providing the description. In Figure 6.22 and 6.23 we can see how the description changes dynamically.

Another factor worth mentioning is that, by restricting the algorithm to not showing the full description until the user is close enough, we can help the legibility of the description by not specifying details that cannot be seen from afar.

Apart from the results extracted from the survey, by being present in some of the surveys done by the users, we could gather information from the users that completed the previous survey. Many of them told us that they felt a significant improvement and that it was much easier to find the target person. Overall, we can see a major upgrade in the algorithm.

Figure 6.22: Start of dynamic change in description

Figure 6.23: End of dynamic change in description

Chapter 7

Individual Work

"Well, there is one advantage to being me. Something you could never imitate. Having you for a friend."

Riku (Kingdom Hearts II)

This project has two authors. Some parts have been developed by both of them, and others have been done individually. The work that each member has done will now be presented.

7.1 Ricardo de la Rosa Vivas

Because the project was going to be made with Unity 3D, first we familiarized ourselves with this tool's workflow. We studied scripting, 3d modeling and basic project management in Unity 3D. Additionally, because this project is base in the previous project (Rabadán and Rodríguez, 2014), we both read and studied the projects documentation and code.

We believed that the code of the previous project was not very legible. Because of this we took upon ourselves to refactor some of the code that we were working with and take care of the new code we wrote.

Using the repository and issues that Daniel created at the beginning we had more control in our project status and the division of work. During the whole project, both Ricardo and Daniel documented their changes and problems in issues at GitHub. This permitted us to have the details of the work done at the beginning of the project and write the documentation more easily.

Firstly, Ricardo was tasked to import the new characters we received from the University; these characters were created by a third party. At this point, Ricardo discovered the clothes problems that are discussed in Section 3.1.3. Having solved this, he investigated how the characters from

the previous project were used and the options we had to replace them with the new characters. This implied an extensive investigation on Unity3D and its models.

Having explore the new characters models Ricardo was tasked to create a script capable of generating the characters dynamically by reading the randomize XML that contained the character's information,. This meant changing the way the XML file was created, mentioned in Section 3.1.1, and creating new C# objects that stored all the information of the clothes that we had at our disposal. This C# object is the one responsible for communication what the natural language description produce to the assets we have in Unity, making the description match the clothes used in the models.

After having the XMLs created in a random and dynamic way, Ricardo was tasked to find a way to generate these characters dynamically. Here is where the dynamic character creation problem, mention in Section 3.1.2, appeared. This predicament consisted in a skeletal issue of the characters, this issue generated when creating the characters piece by piece. Given this major problem, Ricardo develop a series of options that could resolve this issue, among them was the used option. The option was to use the character that was given to with all the clothes on, and undress the character leaving it with the wanted set of clothes, the script capable of doing this was done by Ricardo.

At this point, the refactoring of the code was necessary because many changes were made and the code was becoming illegible.

Having solved the dynamic creation issue, the problem mentioned in Section 3.1.4 arose. This complication consisted on the posture of the characters not being calculated correctly, thus making them pose in unwanted ways. After having done some research on 3D modeling and having gone through all the possible options, Ricardo used a 3D modeling tool to change the name of the joints belonging to the character's bone and created a new set of postures for the new characters, these were 70 new postures.

While Daniel was implementing the first person perspective and scene adjustments, Ricardo focused his effort in increasing the randomizable characteristics when instantiating the characters. This meant adding the capability of creating random female characters and the use of random colors, among others things like changing textures of clothes.

With all prepared for the first survey and while Daniel was deploying the survey, Ricardo started writing the documentation regarding his work so far.

After the first survey was done, Ricardo gathered the information compiled by the survey analyzer and evaluated the first survey. To help the analysis Ricardo took photos of all the scenes starting and ending positions. Having all the information available and having created statistics and comparing tables, Ricardo made the conclusions shown in Sections 4.5 and 4.6.

These were very important because they determined what changes were going to be made to improve the algorithm.

While Daniel was changing the algorithm and preparing the second survey, Ricardo wrote part of the documentations like the Abstract, Resumen, and the Introductions, both in Spanish and in English. At the same time, corrected all the documentation flaws detected by the directors.

Having all the information from the second survey Ricardo took the new photos, analyzed this information and wrote the conclusions shown in Sections 6.4 and 6.5. This is where the changes made to the algorithm, proved that they were a major improvement.

Next, with the project almost done, Ricardo wrote the Future Work and part of the conclusions shown in Section 8.

7.2 Daniel Ruiz Manero

Our first approach was to study Unity's workflow and tools. That basically consisted in studying the basics of C# scripting language and 3d modeling. We read the previous project's documentation (Rabadán and Rodríguez, 2014) and code after we learned the basics.

Daniel prepared the private repository in GitHub and opened a set of issues considering the first steps of the implementation. Those issues were assigned properly to the members of the workteam. We wrote down the emerging problems on the issues during the whole project development.

The previous project code was not very legible. We decided to refactor the code in order to make easier future implementatios. This activity cost an elevated amount of time but it was really useful in future implementations.

During the whole project, Daniel wrote down the documentation parallel to his design and implementation works. That means that every piece of documentation related to Daniel's designs or implementations are written by him.

Daniel implemented completely the first person perspective. He performed the needed scene adjustments and implemented the crosshair in order to recieve input clickcs properly. An example of scene adjustment was the attachment of *Colliders* to the elements of the canteen (tables, columns...). He modified other UI basic elements (the ones related to the first person camera) at this point too. Once the first person perspective was implemented, he desingned and implemented the new H pressed description mechanics. This was considered a new important mechanic in order to proove the algorithm correctness.

After the first person navigation was completely prepared, he wrote down and implemented the first survey instructions because it was thought to be useful for the user. He adjusted the project and the server in order to receive the new generated data (first person related data). On the client side, those

modifications consisted basically to send to the server new variables. On the other hand, PHP server modifications were needed in order to retrieve that data and write it down in the *data.txt* file. He decided and implemented the new data format too. The need of the new data format was a consequence of the new retrieve first person variables.

He compiled and deployed on our University server the project build that was used for the first survey. Daniel implemented the Java survey analyzer application in order to make the survey analysis simpler. The implementation was executed right before the first survey deployment.

During the first survey execution, he wrote down the related work section. That basically consisted on the state of the art, the previous work summary and the Unity related elements.

Next, with the first survey closed an completed, he wrote down some feedback received from the user. He implemented the whole set of detected feedback problems. This problems were really important to solve because user's experience and results might be altered because of these problems. The content reduction of the first survey intructions was performed by Daniel too.

Daniel designed the new Perspective-Meta-Algorithm. He divided the descriptions into sub-descriptions and analyzed them. The analysis basically consisted in what sub-descriptions can represent and where they can appear in a description. The Meta-Algorithm generated description was now considered a sub-description. It was modified to work properly together with the others new generated sub-descriptions. Those simple modifications were implmented by Daniel too. Consequently, he decided the real time relative information obtained during the simulation. The new retrieved data is the one used in the sub-descriptions.

In order to receive this new data, code modifications were needed. The result of those modifications is the Detected script. Daniel designed and implemented the new Detected script completely. He considered the needed data and studied the possible implementations in order to obtain that data in real time. He studied deeply the Unity API and documentation because the implementation has to be as efficient as possible. This information was obtained in real time and could cause a huge computational cost. This was the cause of the efficiency efforts during this implementation.

The Perspective-Meta-Algorithm generates different descriptions according to the scene situation. The elements or input that the algorithm considered when executing are the user, the described person and the static points around the scene. Those decissions were implemented by Daniel too.

During the new algorithm implementation, UI related problems appeared. Those problems were caused by two elements. The first element was the previous code implementation, it was unefficient, redundant and unmaintainable. It was easier to create a new UI than to use the previous one. The

second problem was related to the dynamic generation of descriptions. The descriptions are generated dynamically by the Perspective-Meta-Algorithm. During the curtain scene a first description must be generated dynamically according to the scene situation. It was impossible to easily implement that first dynamic generated description using the previous curtain scene implementation. The conclution was that the previous UI curtain scene can no longer be used and a new one should be implemented. The detection of the UI related problems and the new UI implementation was performed by Daniel.

Lastly, he fill some documentation gaps and checked the documentation a couple of times in order to be sure that it was as good as posiible.

Chapter 8

Conclusions and Future Work

"Building the future and keeping the past alive are one and the same thing"

Solid Snake (Metal Gear Solid 2)

8.1 Conclusions

During the whole process we have made two big tests. The first one consisted in testing the previously implemented algorithm in more realistic situations. The previous tests were more limited due to the fact that the camera was static and could not move. When saying more realistic, this means situations where the visibility of the described is not guarantee and the user have the ability to move around the environment to track and locate the described person easily.

The second big test consisted in creating new descriptions that took advantages of the new implemented mechanics. Therefore, the concept of perspective based descriptions is born. The field of Generation of Referring Expressions has never taken into considerations these parameters which can be used to generate more descriptive and consistent descriptions.

This two big process are basically form by an implementation period, followed by a validation and verification period. Because these algorithms generate descriptions for human to be read, we decided to follow a validation model based in surveys. Both surveys gathered interesting results.

The first stage of implementation was clear. After receiving the new models from the University, we decided that the characters and scenes were going to be generated dynamically, meaning, at run time. At the same time, it was decided that a first persona perspective was to be implemented to test the algorithm. While making these changes, refactoring was made to the code with the aim of having future implementations be easier and quicker. This way, the code increased its quality in terms of legibility.

After finishing the first stage of implementation, the deployment and survey of the application on the server started. For the surveys, not changing 3D simulations were used, this means that the scene were generated once and they do not change when a different users takes the survey. This way the results of the survey are easier to analyze. The deployment was simple and not many modifications had to be made. After the deployment was done, the first survey started. This survey was distributed among people of our environment.

The results of this survey were clear, the users found the described person. 71% success rate was achieved. Nevertheless, we thought that the number of success could be incremented by having descriptions that made use of other types of variables and aspects. This way, we determined that using the distance and visibility between the user and the described persons could be a interesting way to generated our descriptions. Also it was decided to use certain statics points scattered in the cafeteria to generate descriptions that took into account these points. After analyzing the first survey, the second stage of implementation started,

The second stage of implementation has two main activities. The first was to evaluate and solve the problems gathered through the user feedbacks (problems that the user detected while completing the survey). The second activity consisted in implementing the new variables that were going to be taken into account when creating the new descriptions. These were the variables chosen after analyzing the first survey. Therefore, the implementation of the system capable of obtaining these variables frame by frame began. These calculations needed to be made as efficiently as possible because they were executed frame by frame. This system is completely implemented in the script *Detected*. Consequently, the design of the work flow of the new algorithm called Perspective-Meta-Algorithm begun. This new algorithm receives information from the Meta-Algorithm and from the information of the variables already mentioned. The algorithm combines both information and generates new descriptions bases in criteria determined during the design of the Perspective-Meta-Algorithm.

After implementing the algorithm, a new deployment in the same server that the University gave us, was made. The scenes in this new survey are the same as in the previous scene, making the results comparable. The purpose of this survey was to evaluate if the new descriptions allowed the user to find described person the fastest and easiest way. Like the previous survey, it was distributed among people of our environment.

The results of this second survey were very interesting. The number of success between the first and the second survey increased in a 17%, making a total of 88% success rate in the second survey. The feedback received by the users shown that this second survey was easier and quicker to complete than the previous one.

This way, during the project we have implemented an algorithm capable of generation referring expressions in a dynamic way, in situations in which the user is able to move in a 3D environment in a first person camera. Also, this new algorithm generates referring expressions (descriptions) that have a success rate of found characters of 88% in situations described before.

8.2 Future Work

We have been trying to perform all the possible evaluations because the previous project accomplished about generation of referring expressions was no complemented with perspective or user relative descriptions. Besides, there are not many previous studies about this algorithms with this kind of complements. For all that, we expose a set of implementations or tests that can have a lot of interest when studying this field.

The first modification consists of reorganizing or generating the description elements referred to the gender or person as an entity. For example, in the description He is near. *The boy in black near the window.* the idea is if reorganizing the elements referred to the gender or directly to the person the description is more effective when the user finds the described person. A possible reorganization of the previous description is *The near boy in black and close to the window.* According to semantic terms, both descriptions express the same but the second description is shorter becaus is made up of just one sentence thanks to the reorganization of the description elements. In this case, possible descriptions reorganizations should be posed first and if those reorganizations are effective.

Another possible modification is having a specifc character telling the user the description, this way more points of view are added and the description can have characteristic of the descriptor's position, for example *the person with the blue shirt that is in front of me, he is to your right.* By having another point of reference, the space of the target character can be narrow and be found quicker.

An interesting aspect can be to perform a modification where the descriptions are heard by the user, not read. This way, the user can move freely and not deviate his/her sight in order to read the description.

Capítulo 8

Conclusiones y Trabajo Futuro

"Whenever there is a meeting, a parting is sure to follow. However, that parting need not last forever. Whether a parting be forever or merely for a short time... that is up to you"

Happy Mask Salesman (Majora's Mask)

8.1 Conclusiones

Durante todo el proceso hemos realizado básicamente dos grandes pruebas o evaluaciones. La primera consistía en probar los algoritmos previamente implementados en situaciones más realistas. Las pruebas realizadas con anterioridad, parecían más limitadas por el mero hecho de que la cámara era esáatica y no se podía mover. Al referirnos a más realistas, nos referimos a aquellas situaciones en que no está garantizada la visibilidad del usuario y la persona descrita en todo momento y el usuario tiene la capacidad de moverse por la escena.

La segunda gran prueba consistía en crear nuevas descripciones las cuales deberían tomar ventajas de las nuevas mecánicas implementadas. Por todo ello, nace el concepto de descripciones de perspectiva o relativas al usuario. El campo de generación de expresiones de referencia nunca ha tomado en consideración estos parámetros los cuales pueden servir para generar descripciones más explicativas y consistentes.

Estos dos grandes procesos o evaluaciones básicamente están formados por una etapa de implementacón seguida de un periodo de validación y verificación. Al ser algoritmos que generan descripciones que deben ser leídas por personas, optamos por un modelo de validación basado en encuestas.

La primera etapa de implementación fue bastante clara. Primero tras recibir los nuevos modelos por parte de nuestra Universidad, decidimos que las escenas y los personajes se generasen de forma dinámica, es decir, en tiempo

de ejecución. Al mismo tiempo, se decidió implementar la perspectiva en primera persona para ver como se comportaba el algoritmo. A la par que estas implementaciones, se decidió aplicar un proceso de refactorización al código para que las futuras implementaciones fuesen más rápidas y sencillas de implementar. Del mismo modo, el código incrementó su calidad en términos de legibilidad.

Tras acabar el primer periodo de implementación, empezó el despligue de la aplicación en el servidor de nuestra Universidad. Para los despliegues o las encuestas, se han usado simulaciones 3D que no cambian. Es decir, se han generado una vez las escenas y estas no cambian entre ejecución y ejecución. El despliegue fue muy sencillo y no hubo muchos elementos que modificar. Cuando se terminó el despliegue, se puso en marcha la primera encuesta. Esta encuesta se distribuyó entre personas de nuestro entorno.

Los resultados de esta encuesta eran bastante claros. Los usuarios encontraban de forma muy correcta a la persona descrita en la escena correspondiente. Se obtuvo un porcentaje de aciertos del 71 %. Sin embargo, creíamos que el número de aciertos se podía ver incrementado usando descripciones que se refiriesen a otro tipo de variables o aspectos. De ese modo, determinamos que usar la distancia y visibilidad entre el usuario y la persona descrita podía ser muy interesante para generar nuestras descripciones. También se determinó usar ciertos puntos estáticos esparcidos en la cafetería para generar descripciones en función de la proximidad de la persona descrita a esos puntos. Tras finalizar el análisis de la primera encuesta, empezó el segundo periodo de implementación.

El segundo periodo de implementación se basaba en dos actividades. La primera consistía en evaluar y solucionar los problemas recibidos a través del feedback de los usuarios (problemas que observaba el usuario durante la realización de la primera encuesta). La segunda actividad básicamente consistía en implementar las nuevas variables que se iban a tener en cuenta a la hora de generar nuestras nuevas descripciones. Estas variables son las que se decidieron tras analizar la primera encuesta. Por todo ello, se empezó a implementar el sistema que obtuviese todas esas variables en tiempo de ejecución cada frame. Esos cálculos se debían de realizar de la forma más eficiente posible ya que se ejecutaba cada frame. Este sistema está completamente implementado en el script *Detected*. Por consiguiente, se empezó a diseñar el flujo de trabajo del nuevo algoritmo llamado Perspective-Meta-Algorithm. Este nuevo algoritmo recibe la información correspondiente del Meta-Algorithm y la información de las variables descritas anteriormente. Combina ambas informaciones y genera nuevas descripciones en función de una serie de criterios determinados durante el diseño del Perspective-Meta-Algorithm.

Tras implementar el algoritmo, se realizó un nuevo despliegue del proyecto en el mismo server que nos suministró la Universidad. Las escenas de

esta nueva encuesta son las mismas que las de la encuesta anterior para poder realizar comparaciones. El propósito de esta encuesta simplemente era evaluar si estas nuevas descripciones permitían al usuario encontrar a la persona descrita correctamente y lo más rápido posible. Del mismo modo que la segunda encuesta, se distribuyó entre las personas de nuestro entorno.

Los resultados de esta segunda encuesta también eran muy interesantes. El número de aciertos entre la primera encuesta y la segunda se vió incrementado en un 17 %, es decir, llegamos hasta un porcentaje de aciertos del 88 %. El feedback recibido esta vez por los usuarios exponía que esta segunda encuesta fue más fácil y rápida de realizar que la primera.

De este modo, durante el desarrollo de este proyecto hemos implentado un algoritmo generador de expresiones de referencia de forma dinámica (en tiempo real) en situaciones en las que el usuario es capaz de moverse por un entorno 3D con una cámara en primera persona. Además, este algoritmo genera unas expresiones de referencia (descripciones) las cuales tienen un porcentaje de personas correctamente encontradas (aciertos) del 88 % en las situaciones descritas anteriormente.

8.2 Trabajo Futuro

Hemos intentado siempre realizar todas las pruebas posibles ya que el trabajo previo realizado sobre la generación de expresiones de referencia no estaba complementado con descripciones de perspectiva o relativas al usuario. Además, no existen muchos estudios previos sobre esos algoritmos con este tipo de complementos. Por todo ello, exponemos una serie de implementaciones o pruebas que pueden tener mucho interés a la hora del estudio de este campo.

La primera prueba consiste en ver como podemos organizar o generar los elementos de las descripciones referentes al género o a la persona como entidad. Por ejemplo, en la descripción *Está cerca. El chico de negro que está al lado de la ventana* la idea es saber si reorganizando los elementos que se refieren al género o directamente a la persona la descripción es más eficaz a la hora de que el usuario encuentre a la persona. Una posible reorganización de la descripción anterior es la siguiente *El chico de negro está cerca y al lado de la ventana.* En términos semánticos, ambas descripciones expresan lo mismo pero la segunda descripción es más corta ya que solo está constituida por una oración gracias a la reordenación de los elementos de la descripción. En este caso, primero se debería plantear qué posibles reorganizaciones puede tener una descripción y si esa reorganización es eficaz.

Otra posible prueba consiste en tener un personaje fijo que le diga al usuario la descripción. De esta manera, se añaden más puntos de vista y la descripción puede utilizar carácterísticas espaciales del emisor. Por ejemplo *La persona con camisa azul que tengo delante de mí, él está a tu derecha*, teniendo otro punto de referencia, el lugar donde puede estar la persona

descrita se reduce, agilizando su busqueda.

Un aspecto interesante que se puede abarcar es el de que las descripciones fuesen escuchadas por el usuario en vez de ser leídas. De esta manera, el usuario puede moverse con tranquilidad y no desviar la mirada de su entorno para leer la descripción.

Bibliography

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

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

Hervás, R. (2009). *Referring Expressions and Rhetorical Figures for Entity Distinction and Description in Automatically Generated Discourses.* PhD thesis, Universidad Complutense de Madrid, Spain.

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

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

Pechmann, T. (1989). *Incremental speech production an referential overspecification.* Linguistics.

Rabadán, A. and Rodríguez, T. (2014). *Generating Referring Expressions in a 3D Environment.* Universidad Complutense de Madrid, Spain.

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

Reiter, E. and Dale, R. (1995). *Computational interpretations of the gricean maxims in the generation of referring expressions.* Cognitive Science.

Unity-Technologies (2015a). Unity manual. Online Distribution. http://docs.unity3d.com/Manual/index.html.

Unity-Technologies (2015b). Unity scripting api. Online Distribution. http://docs.unity3d.com/ScriptReference/index.html.

Winograd, T. (1972). *Understanding natural language.* Academic Press, New York.

For the both of us... you're gonna... live. You'll be... my living legacy.
My honor, my dreams... They're yours now...

Zack Fair (Final Fantasy VII: Crisis Core)

I'll be going now. I'll come back when it's all over.

Aerith Gainsborough (Final Fantasy VII)