Mapping 3D Character Location for Tracking Players’ Behaviour

João Emílio Almeida, Rosaldo J. F. Rossetti
Artificial Intelligence and Computer Science Lab.
Department of Informatics Engineering
Faculty of Engineering, University of Porto
Porto, Portugal
{joao.emilio.almeida, rossetti}@fe.up.pt

António Leça Coelho
National Laboratory of Civil Engineering
Lisboa, Portugal
alcoelho@lnec.pt

Abstract— Serious Games are increasingly used as a tool for various applications contrary to the traditional entertainment purpose. Many game engines are available, and Unity3D is another example that presents some features such as rapid prototyping and an easy learning curve. The 3D space where action takes place is sometimes hard to map into a logical memory structure providing flexible access to that information. The problem of tracking players’ path as well as their decisions in 3D environments arises when there is no previous knowledge of the scenario representation and the creation of a memory data structure poses an extra effort for the modeller. The proposed solution in this paper, albeit simple, is a straightforward way to track the location of the video game character and map it when he/she passes certain limits. This mechanism proved to represent a key step forward, addressing the important issue of tracking the decision-making process of players for future analysis and behaviour elicitation.

Keywords: serious games; behaviour analysis; decision-making modelling; 3D character tracking.

I. INTRODUCTION

The advent of video games has fostered the implementation of Serious Games (SGs) with various goals other than entertainment. SGs have been used for many purposes including training, education among others, but also for social sciences and psychological experiments [1]. As McGonigal stated [2] it takes a clear goal that we can achieve and direct feedback for work to feel positive and even pleasant, and games excel at providing both. In fact, the author even refers that by only bestowing our best efforts into overcoming a game’s challenge, we feel more rewarded.

Unity3D is a successful platform used worldwide for the development of video games [7]. Our team has used it to the development of Serious Games to train occupants in the evacuation procedures of a building, under the presence of fire or another hazardous situation [8]. For this purpose, we have a 3D scenario representing a building, having the player to navigate through pathways until reaching the outside where he/she is safe. The model is a 3D representation of a building; whether real or just simulated, with a certain degree of complexity, meaning the distribution of spaces is rather tricky to be easily assimilated by any unfamiliar user placed somewhere, and having no previous knowledge of the architecture. Our aim is to reproduce a recurrent situation when a visitor of large buildings such as office building, hospital, shopping centre, gets lost in the vast corridors and hallways; if a fire alarm sounds, he/she has to evacuate as quickly as possible, following the emergency signage, or trying to find the best and safe way out.

Unity3D provides an excellent framework for developing such scenarios, based on existing architectures, having the capability of importing from various 3D formats. Nevertheless, although it is possible to import CAD models into games, this process has several drawbacks and it is not that easy to automate [9]. For instance, when the pathway of the character moving around in the tri-dimensional space is to be tracked, so that player’s tactic and strategic decisions can be analysed, there must be an efficient mechanism that allow us to easily memorise and replicate a wide range of actions reflecting the adopted behaviour. Indeed, there is no such a way of making a direct connection with the spatial location of the \((x, y, z)\) coordinate system usually implemented in game engines and some specific places inside the building, where the player rovers towards the exit. Given this problem, the use of landmarks seems to be a natural candidate for a pragmatic solution. For instance, imagine the game character is placed in a certain room of a large building; the player is instructed to find a route to exit the building. How can we track her route options made during the game? Asking in a different way, is it possible to map the coordinates of the player position within the actual building? If the scenario is set to a hotel with a main corridor and five bedrooms on each side, the question is how to map every one of these compartments to a dataset in order to know where the character is at each moment during the game, making the correct correspondence to the real building, if it exists. It may seem trivial at a first glance, however it reveals to be quite tricky and time-consuming whenever the prototyped model is intended to be simple and should just roughly represent reality; another situation in which such a problem arises is when building models are synthesised automatically through techniques such as procedural modelling.

In this paper a rather simple but straightforward and efficient mechanism to easily track movements of videogames characters in a tri-dimensional space is presented, using Unity3D box colliders to generate landmarks. The itinerary followed by the player whose character navigates over the
environment is recorded for future analysis. This provides a means for both a syntactic and semantic structure useful for mining behaviour afterwards.

The remainder of this paper is organised as follows. Section 2 presents some background and related work in the field of serious games. Section 3 is used to introduce the proposed methodology to track players’ choices when navigating in a 3D environment, whereas the experimental setup and tasks to accomplish such realization are detailed in Section 4. We finally draw some conclusions and point out future work directions and further developments in Section 5.

II. BACKGROUND AND RELATED WORK

A. The Serious Games Concept

Serious Games has gained a great prominence in the Digital Games field within the last years, by using high-definition graphics and state-of-the-art appealing software. It presents a great potential as a tool to be used for other purposes than mere entertainment. Applications have a wide range of domains [15], naturally including social simulation, where data collection of players’ attitudes can be later used for statistical analysis, and behavioural pattern recognition. Contrary to the primary purpose of entertainment in traditional digital games, SGs are designed with a more serious purpose with respect to the outcomes reflected in changes to the player behaviour [1, 2].

A game is an artificially constructed, competitive activity with a specific goal, a set of rules and constraints that is located in a specific context [3]. SGs refer to video games whose application is focused on supporting activities such as education, training, health, advertising, or social change. Freitas [4] said that benefits from combining them with other training activities include: i) the learners’ motivation is higher; ii) completion rates are higher; iii) possibility of accepting new learners; iv) possibility of creating collaborative activities; v) learn through doing and acquiring experience.

Other aspects that draw video game players’ attention are fantasy elements, challenging situations and the ability to keep them curious about the outcomes of their possible actions [5]. Some authors [6] classify SGs in five categories: Entertainment, Advergaming, Edumarket Games, Political Games and Training and Simulation Games.

B. Using Unity3D to Implement Serious Games

The Serious Game used as example in this research is based on the Unity3D game engine [10]. This game engine was selected due to its characteristics, among them [8]: i) powerful graphical interface that allows visual object placement and property changing during runtime (especially useful to rapidly create new scenarios from existing models and assets and quick tweaking of script variables); ii) the ability to develop code in JavaScript, C# or Boo; iii) simple project deployment for multiple platforms without additional configuration, including for instance the Web (which makes it possible to run the game on a Web browser).

C. The game genre: First Person Shooter

First Person Shooters (FPSs) are characterised by placing players in a 3D virtual world which is seen through the eyes of an avatar. This attempts to recreate the experience of the user being physically there and exploring their surroundings. When playing, user has the feeling of being actually on the location site, moving around, and giving the best possible sensation of immersion.

There are other devices and techniques, such as CAVE and Virtual Reality that provide a better feeling of immersion to the player. However, simply using SGs is far more economic and feasible, without having to use expensive hardware and software.

The controls for this game follow the common standards for the FPS genre, using a combination of keyboard and mouse to move the player around the environment. The complete action mapping is as follows:

- **Mouse movement** - camera control, i.e. where the player is looking at;
- **W** - move forward;
- **S** - move backwards;
- **A** - move to the left;
- **D** - move to the right;
- **Space bar** – jump.

III. TRACKING PLAYERS’ DECISIONS

The aforementioned problems of mapping the user or character location within the corresponding space in the building can be resolved by the use of invisible objects.

For the sake of a previous explanation, consider a 3D synthetic digital environment, where objects can have physical characteristics, such as mass, making them insurmountable; for instance, the plan that represents the floor where characters are walking must provide a barrier to prevent them from falling. Walls must also be detected in such a way that the game subject does not pass through it. Unity3D physics engine implements this automatically providing that the objects are defined as solid structures. When importing an architecture file describing a building, all elements such as the façade, floor, walls, windows, are defined as solids. So, when a first person shooter character is placed in the building, using the movement controls, player can travel inside as if it was wandering around, providing the feeling of being actually immersed in the environment.

The problem arises when the location of the character roved by the player is an important issue. When the tri-dimensional scenario is imported from a CAD file into the game engine, the meaning or semantics related with the site, such as the floors or compartments names, is lost, if ever was present. And this data is needed for knowing where the character is moving and in what direction.
A. Tracking position of character using polygons

The issue of mapping the location of a player to the existing building is only possible by previously knowing all the coordinates of the rooms and having a memory structure with that information (Table 1). The set of coordinates defines a polygon. When the actual coordinates of the character are inside one of the polygons, then the exact location is set and it is possible for the game to track the rooms visited by the player. This data can be saved and analysed later.

<table>
<thead>
<tr>
<th>Floor</th>
<th>Compartment data</th>
<th>Boundary set of points</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Hall entrance</td>
<td>p1, p2, p3,…</td>
</tr>
<tr>
<td>0</td>
<td>Living room</td>
<td>p1, p2, p3,…</td>
</tr>
<tr>
<td>0</td>
<td>WC</td>
<td>p1, p2, p3,…</td>
</tr>
<tr>
<td>1</td>
<td>Bedroom 1</td>
<td>p1, p2, p3,…</td>
</tr>
<tr>
<td>1</td>
<td>Bedroom 2</td>
<td>p1, p2, p3,…</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

To create such a memory structure as the one presented a considerable amount of time and effort is needed, to gather all the necessary data. An automatic procedure to assemble this information dataset could be devised and implemented. Of course smaller datasets will not represent so much trouble to be built; however, the amount of data is expected to increase drastically in both dimensions, namely in number of rooms descriptors and reference points for complex compartmented buildings, such as hospitals and shopping centres, with hundreds or even thousands of rooms.

However, there is a much simpler way of solving this problem, with less effort, and achieving the same result. The methodology that we propose in this paper, a simple but effective one, is described hereafter.

B. Tracking position of character using colliders

As referred previously, in 3D synthetic environments, in order to reproduce with a certain degree of similitude the real world in a computer, objects must have the same behaviour they have in reality. For instance, solid objects must not be crossed or transposed, for which some sort of collision and the correspondent physical effect must occur. Engine games provide the tools to achieve this effect, by assigning material characteristics to objects, and calculating collisions with other elements moving through the scenario. When they collide, the movement stops, like when a person bumps into a wall or closed door. This reaction will provide a sensation of reality and immersion to the player, such as it would happen in real life.

It is also possible to assign an action each time one object collides with another. This is how, for instance, automatic doors are implemented in games. A boundary object is defined, surrounding the character, e.g. a circle or a cylinder, to simplify the collision calculus. When the character is moving, this cylinder is tested against existing objects, and if a collision is detected, that information is sent to the routine that controls the character movement. A trigger is set and some pre-defined actions can be taken. Using the automatic door example, when the character cylinder collides with the door, an animation procedure can be invoked to slide or rotate the door, allowing the character to transpose that obstacle. Another possibility is the player to use one command, with the mouse or keyboard, to order that door opening.

This is how we implemented in our test game the command to open doors. Each time a collision with a door is detected, and a Boolean state associated with that door indicating if it is open is set to “false”, if the user press the correspondent key, assigned to command the door movement, then the following procedures occur:

- Door animation procedure is invoked, and the correspondent object is turned 90° over the hinges axis, at a certain speed; a sound can also be heard, indicating that operation was executed successfully;
- The Boolean variable that has the door status (open) is set to “true”.

Fig. 1 shows one example of box colliders (shown as yellow wireframe) near a door, o detect if the character is near enough to be able to open it. The limits are not visible when running the game, but each time the character crosses the boundaries of the box, an event trigger is set allowing some code to be executed.

![Box collider inside a room.](image)

To implement some sort of logic regarding the location of the character, a similar solution can be used. By creating an invisible object, like a box, in the places where the location of the character is needed, the game can track the movement by recording the collisions. Each time the player directs the movement of the character towards the place where the invisible box is located, an event is triggered and the position can be recorded, with some other key elements, like time. Using this simple concept, it is possible to create a data log with the places where the character has been into.
If the aim of the game is to have the character perform a series of tasks in several places, to check each one of these activities, it is possible by creating invisible box colliders in the required locations. Each time the character bumps into that invisible object, a trigger sets data to be recorded with the id of the object and time stamp. Later, the path of the character can be recreated and also with the time stamp, velocity can be inferred. Using this technique, and the adequate dataset, it is possible to create a chart of results to compare the performance of various players.

IV. EXPERIMENTAL SETUP

A large three floor building was used as the scenario for data acquisition. The 3D model was imported from an Autodesk’s Revit file into the Unity3D framework. A character using the FPS game genre was created.

The player is instructed to wander around the building. When colliding with the invisible colliders placed in key locations, an event is triggered, and in some situations questions are posed. Answers to those questions are saved in a file with the timestamp. Data collected is saved in a comma separated value (csv) file for later analysis and processing, using a spreadsheet system or statistics package. For the purpose of this paper, the final goal is not important, so the discussion about the questions and positioning of the aforementioned landmarks is not detailed here. The important issue to retain is the methodological approach used to gather information and connect it to a 3D environment.

![Figure 2: Box collider outside exit door.](image)

One of the goals of this experiment relies on testing the ability of the player to find the fastest exit to the outside of the building. Players do not know the architectural layout a priori. The game starts with the player inside a predefined room. The player must then transverse the building and get to the outside as quickly as possible, choosing from one of the possible exits. Several emergency signs are in place in order to help identify the nearest exit. When the player finally leaves the building, by crossing one of the limits, the selected exit is recorded (see Fig. 2). Also, some invisible colliders are set in key points to track the player’s options. At the end, a final inquiry will ask some questions in order to help understanding the choices made by the player, whose answers might be related to the itinerary chosen by the player and identified by the tracking approach presented in this paper.

Two other events are triggered, namely i) when the player enters the room which will make the fire alarm sound and ii) afterwards, when trying to exit the building, if the player chooses to use the elevator a warning message is presented. In both situations, the location of the player within the building is determined and saved using the colliders instead of having to perform more complex calculations to compare the character position with all the existing rooms. The memory structure with all the locations of the building’s rooms is also expendable.

Using this technique there are some obvious benefits. During the game development, it is far smoother to use invisible colliders instead of the traditional tracking process. The game engine provides all the necessary tools and further programming is avoided. The traditional approach to identify the itinerary of a character in a game would involve identifying the sequence of coordinates to be covered, constituting a time consuming procedure. This is easily accomplished with our approach as this is now just a matter of replacing the box colliders on the correct positions throughout the route, whenever the itinerary changes. Additionally, debugging is also easier and the probability of errors is lower.

V. CONCLUSIONS AND FUTURE WORK

This paper introduces a simple but effective and straightforward technique to track the location of the video game character and map it when he/she passes certain limits. Although this solution might seem rather simplistic, it perfectly solves the important problem of tracking decisions for future analysis and behaviour elicitation. An example is presented using the popular Unity3D game engine. The alternative to this would be to manually create a data structure with all locations to be tracked and their coordinates, for later testing continuously if the character was colliding with any of those regions. A longer time to develop it would be needed, increasing the complexity of modelling and testing phases. This also raises the possibility of errors and increases the complexity of the debugging process.

Future work will expand the use of box colliders in 3D scenarios creating an automated procedure to include the invisible collider in place and associate it with the event to be recorded. Our aim is to improve productivity when importing 3D architectures from other applications, such as Revit, into Unity3D, using built-in features of the game engine to assist the modeller and shortening the modelling phase. Experiments to emphasise this simple technique and demonstrate its validity will be performed. More research on various ways to explore potentials of this mechanism of tracking location of game characters for allowing a posteriori analysis of different behaviours (past decision-making) is necessary. This is part of a wider project aiming to analyse players’ behaviour in a
particular application domain, namely the study of human behaviour when evacuating buildings in emergency situations [11, 12, 13, 14, and 16].

ACKNOWLEDGMENT

This project has been partially supported by FCT (Fundação para a Ciência e a Tecnologia), the Portuguese Agency for R&D, under grant SFRH/BD/72946/2010.

REFERENCES