ACID RAIN: a virtual environment for the entertainment

Andrea Monacchi*
MSc student at University of Camerino

David Sorrentino†
MSc student at University of Camerino

Abstract

In this report we describe the project Acid Rain, a virtual environment for the entertainment. The Acid Rain prject wants to reach a variegated age target with the aim to entertain but also push it to the reflection and consideration of the risks which are behind the nuclear technology.
We started with a review on similar works to clarify our strong points respect to them. Then we passed to the design stage to define our main goals and finally we selected some of these as expected results. We also chose how to achieve these goals defining our development methodology.
In the next part we report some technical solutions adopted in the project and in the end we show achieved goals and some further and future improvements to make.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Radiosity

Keywords: virtual environment, earthquake, nuclear fallout, corruption

1 Motivation

The aim of this project is creating a virtual environment to entertain users addressing the problem of radioactive contamination and the corruption between companies and governments, which unfortunately is a reality. The usage of a virtual environment allow us to reach also the youngest target in order to sensitize it about these current topics. Therefore the project has also an educational aim since it wants to remind to users the risks which are behind some technologies managed by big companies. The environment should be as accurate as possible in order to guarantee a strong sense of presence to the user. The user should be also surprised by an unpredictable and complex plot made of suspence, turnups and a surreal atmosphere.

2 Related Works

To follow our aims, firstly, we looked for similar projects to understand their strong points. We found a similar scenario in the Fallout series, where we encountered a post apocalyptic setting after a nuclear world war (and the subsequent nuclear fallout) which caused the depopulation of the country but also the contamination of the environment and the mutation of its forms of life.
This is an RPG game. Our project would like to be more intense and reflexive, using a more detailed story which can better involve the user and give him a strong sense of presence. In fact, our aims are also educational, since we want to remind the risks behind the nuclear technology.
So, the Fallout series is similar from the scenario point of view, but the genre is completely different.
Another title we wish to mention is Heavy Rain, a psychological thriller where the user has to investigate to catch the Origami Killer. Here, the user has to interact with the objects and the non-player characters of the scene, in order to progress the story. In every scene the user plays one of the four different playable characters. When the player gets close to an object or another character is showed a context-aware interface which shows the controls he can perform in that situation. There are also time-dependent tasks which the user can perform in his context but respecting certain time limits.

3 Approach

In this section we present the scenario and the archetypes involved. Then, we report the design stage where we divide the project in smaller blocks and the implementation section in which we describe some technical solution adopted in our project.

3.1 Scenario and archetypes

The user starts his experience writing a message on his mobile phone. He is communicating his arrival in Varna and his car crash caused by an earthquake. He is a reporter and he was sent by his boss to investigate about strange things which are happening there after the seismic swarm. In fact, the frequent earthquakes have caused problems to the close nuclear plant and someone started to speak about a possible contamination, which made mutate the close village inhabitants. The village is now desolate and almost deserted. You have to ask who remained, investigate about these rumors and try to clarify this foggy situation.
To make the environment more realistic we decided to put all the characters into it since the beginning. They are in an idle state where they present theirselves without anticipate their future task. In this way the user can interact with them without jump any part of the story.
We defined two different epilogues (figures 1 and 2), reachable through four different paths. In the first case the user solves the main goal about the mutation rumors and he shows the guilty on his newspaper. In the second case, instead, he decides to go away without going into further investigations. We also connected in some points the paths, in order to twist the story and make it more complex.
We have six different actors: a farmer and his granddaughter, a grave digger, a mayor, a watchman and a sailor.
According to the Vogler's analysis we can notice some archetypes in our plot. Firstly the hero, who is represented by the user. The mayor represents a mentor the user can use to go away. The Mayor is also a shapeshifter, since if the user choose to stay for investigating, the mayor uses the rumors as an information to push his departure. The mayor is also the shadow because he is guilty of the

*e-mail: andrea.monacchi@gmail.com
†e-mail:sorrentinodav@gmail.com

Figure 1: *the first final*

Figure 2: *the second final*

disaster caused by the nuclear plant, since he agreed to the buildings by getting money. The farmer and his granddaughter, instead, have the task to take the user in the story, giving him first informations and allowing to him to practice with the commands. The gravedigger, also, covers a marginal role because he is not involved in the disaster but he assures you the rumors are false.
The watchman, instead, is an ally because he helps the user to give to the mayot the documentation box (false or not, according to the user's choice).
At last, the sailor, who is the threshold guardian, since he brings away the user for money, independently from his choice for the box. That is an important point where the story ends.

3.2 Design

The design phase started with a division of the tasks. We distinguished the core features from the optional ones.
The core features are essential for the good execution of the game. In this respect we should create and manage the terrain, the user interface, the behaviour of actors, objects and collisions. To cover these points we chose a top-down approach.
The optional features are indeed improvements and surplus values we should add to our project in order to improve the sense of presence. They are appearance matters related to the environment dynamism. We should add shaders (in particular on the sea waves), use a realistic weather (an heavy rain), but also use sound effects and animations on characters and objects.
To these points we recognised also special features we should cover as an improvements of the main ones, but which characterize our project. For the interface we chose to always show the current goal and list all the possible interactions with actors and objects in a sort of context aware label. Therefore the user should be able to pick up objects and use them into the environment in order to improve the sense of parallelism and the dependence of events.
We chose to work firstly on the core features, then on the special ones and finally on the optional ones.

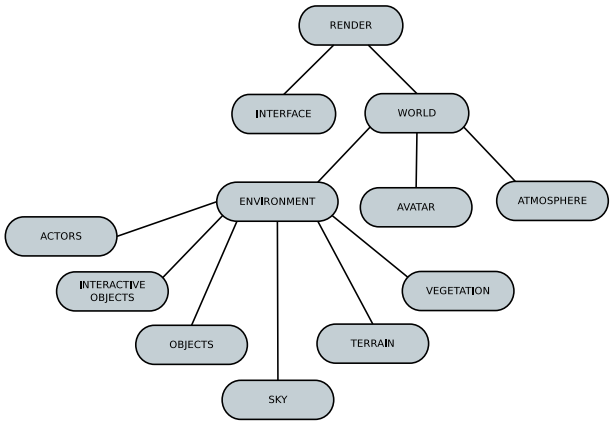

Figure 3: *The Scene Graph*

3.2.1 Scene Graph

During the design phase we fixed the macrostructure of the scene graph as shown in figure 3.

3.2.2 Expected Results

We expect to cover the main tasks to allow the execution. Then we would like to work on the special and optional features. So, we chose firstly to work on the environment, adding the terrain and the collisions between it and the avatar. Then we chose to focus on the user interface and finally add objects and actors, and define their behaviors and interactions in order to guide the users' experience, providing to him some goals to achieve.

3.2.3 Development Methodology

We decided to work together exploiting the technique of the "pair programming". It is an agile software development technique in which two programmers work together at one workstation. While one typed in code, the other reviewed each line of code as it is typed in. In addition, we switched roles frequently. This approach ensured greater homogeneity to the whole project and guaranteed a consistent saving of time avoiding long periods of testing.

Figure 4: *push to talk*

3.3 Implementation

3.3.1 Interface and commands

Interface is one of the most important feature of the project. We chose to make it lightweight and simple. We rejected the concept of command list because we don't want to make the user keep in mind all the keys. We thought to use a context aware label on the right top of the screen to show in every situation the list of actions available to use. Every time a user approach an actor, he can choose to speak to him by pressing the space button on his keyboard. Similarly, when he get close to an object, he can decide to pick it up by pressing the same button. In this way we limit the number of key to use and we can always remind to the user the actions available.
We needed to define three different kinds of context, one for the normal state of the user where we list his inventory and actions to move him into the environment, one for when he approaches an actor (which lists interaction commands) and finally one for the picking of the objects.
The context aware commands label is refreshed every time an action is added so that for example when he picks up a map he can use it by pressing the 'm' button. This means we could have problems to list all the commands in long time game experiences (where we could pile up too many commands) but for our aim it is the best choice.
All the commands we chose are placed on the left side of the keyboard. We used the 'WSAD' combination to move the avatar on the terrain and the mouse pointer to direct his view and allow a customized movement.

In addition, we chose to show always the next goal of the user using a text label on the left top of the screen. In this way the user is guided in his game experience but he is free to explore the environment, so that he does not feel it too much explicitly constrained.
For the dialogs we used a centered bottom label and we blocked the user until the end of the conversation. In fact, we used the 'c' button to continue the dialog when it is too long to be shown on a single line. In this way the label preserves its dimension height for every kind of text and it avoids to disturb the user view (modeless). So, when the user approaches someone, he can talk to him by pressing 'space' and continue the conversation answering to questions or simply going ahead with the button 'c'. When the conversation ends the context is refreshed and the user commands unblocked to allow him to leave.

Figure 5: *push to pick up*

3.3.2 Avatar management

Regarding the perspective, we chose a first-person perspective to ensure a better navigation and interaction. Moreover, this type of perspective makes the user feel more involved into the game.
To guarantee that the avatar is always at the right height respect to the height of the map, we update the height of the avatar every frame, depending on the height of the map. In particular, to know the precise height of the position in which the avatar is, we used the method getElevation belonging to the class GeoMipTerrain.
Instead, to manage the collisions with the edges of the map, we used a bitmap image containing only black parts and white parts. The black parts represent the parts to which tha avatar can go; the white ones, instead, represent the parts to which he can not go. After creating the above-mentioned bitmap, we transformed it into a matrix of coordinates containing the value 0 or 255 (black parts or white parts). Therefore, when the avatar is changing his position, by means the above-mentioned matrix it is possible checking if the future position is allowed. If it is, the avatar can actually change his position.

3.3.3 Environment

Terrain Regarding the terrain we stretched a texture image onto a heightmap of the same dimension. For the terrain generation we used the class GeoMipTerrain because it guarantees better performances than highTesselator. The main feature of the terrain used is the height difference, that ensures a good occlusion enhancing the presence and lightening the rendering phase.

Vegetation Regarding the vegetation we designed a class that has the aim to place and manage the vegetation on the entire map. To do it, the class in question takes in input a given type of plant and a list of coordinates in which that type of plant will be placed. After placing the plants, the class manages the collisions between them and the avatar adding a collision sphere around each plant. For each type of plant, we loaded the model and applyed it to a series of nodes using the instancing technique.
The nodes which represent all the plants are subnodes of the node vegetation, that in turn is a subnode of the node environment.

Atmosphere Regarding the atmosphere we chose to use a dark ambient light in order to enhance the presence and make the discovery of the map more interesting. Moreover, we chose to use a thick fog to cover the distant parts of the map and lighten the rendering phase. About the sound, instead, we chose to use just a few sounds (an ambient music, footsteps and foliage sounds) because in our game the silence plays a essential role.

Figure 6: *Loading*

Actors We designed a class that has the aim to place and manage all the actors within the environment.
Regarding their behaviors, we defined the actors as finite state machines, of which each state corresponds to a behavior. The main behaviors are three: looking at the protagonist when he gets close, starting a dialog when the protagonist interacts with them, returning in an idle state when the protagonist walks away.
Regarding the dialogs, in turn we defined them as finite state machines, of which each state corresponds to a different speech. In this way, to create a new character is enough specifying the model and the position on the map, and finally defining the dialog that will characterize it. The special feature of the actors is that they react to the story even if the protagonist does not interact with them. In practice, by means of a system based on the events, they always know the part of the story in which the protagonist is, and update their state according to that part.

Interactive objects Regarding the interactive objects, we designed a class that has the aim to place and manage them within the environment.
As for the actors, we considered the interactive objects as a finite state machine. The main states are two: the object is static and the object is dynamic and pickable. Also in this case the special feature is that the interactive objects react directly to the story. In practice, by exploiting the same system based on the events of the actors, the interactive objects know the part of the story in which the protagonist is, and change their state according to that part. Therefore, in a certain part of the story, the protagonist will be able to interact with objects with which he could not interact before.

3.3.4 The Loader

Since we have a rather detailed environment, its loading requires some time. Therefore, we chose to insert a loading stage before running the game in order to inform the user about what is happening during the loading, avoiding that he remains alone with a black screen (fig. 6).

4 Results

We achieved all the expected results (environment, collisions, story) and we paid particular attention to the special features (interface, interaction with objects and event flow) so that it was concretely working but also respecting our minimum requirements to guarantee the presence.
We added also some sound effects and musics but we did not cover the other optional features.
Moreover, we used the object oriented paradigm which allows to us modularity and code maintainability. We could make some level for the game, using a chain of events where we instantiate different "World" class, each of these managing a different world three (figure 3).

5 Future Work

We should complicate the event flow of the story so that it was more challenging and intriguing. In this way it would be also longer.
We need to cover the optional features in order to complete the project. In particular we should add animations to our characters, since actually they seem a bit much cold. Furthermore we should improve the appearance and the dynamism of the environment adding sound effects and localized sounds (for instance the sea breeze) but also the weather and in particular the rain.

References

LOMBARD, M., AND DITTON, T. 1997. At the heart of it All:The concept of presence. *JCMC 3*, 2 (Sept.).

VOGLER, C., AND MONTEZ, M. 2007. *The Writer's Journey: Mythic Structure for Writers*. Michael Wiese Productions.