Computer Graphics Unity Project - The SpaceMan

Abstract— **The purpose of this project is to explore creativity and provide a forward thinking about computer graphics and the requirements that are needed to implement a game. This report will detail the features and techniques to implement a 3D game with a character capable of moving and interacting with the environment of the game.**

Keywords - Animation, 3-Dimensional, Frame, Image, Digital, Transform, Character, Platform.

I. INTRODUCTION

Today computer graphics is widespread and many tools have been developed to visualize data. Computer generated imagery can be categorized into several different types: two dimensional (2D), three dimensional (3D), and animated graphics. Computer graphics can be seen as faking reality, in other words the ability of creating a new reality that replicates real environments. To create objects in 3D there are 3 essential techniques that are modeling, animation and rendering. The way we represent objects and how to create them is through modeling, while for controlling the way the objects move is through animation and for creating images it is used rendering.

The 3D game that is explored in this report is called ”The SpaceMan”, which takes place in planet Venus and it is constituted by one character that is a spaceman, which main goal is to go through a path of lava with platforms to get to his spaceship. The platform utilized to create SpaceMan was Unity3D that supports 2D and 3D graphics, drag-and-drop functionality and scripting using C#.

II. PROJECT OVERVIEW - FEATURES AND REQUIREMENTS

First to meet the demands of a 3D game, it will be necessary to have some features for the game to work and look at least similar to real physics of the real world. Heres some features, requirements and techniques that i have used through Unity in the project:

A. 3D Objects

Typically, an example of the object to be recognized is presented to a vision system in a controlled environment, and then for an arbitrary input such as a video stream, the system locates the previously presented object. Instead of using 2 system coordinates like 2d, 3d uses a 3 system coordinates (x,y,z), which is important to cover that not all game engines or 3d software use the same coordinate system. With Unity axis-Y is up and down, axis-X is left and right and axis-Z is in and out.

The game objects present in the game are cubes, rectangles, cylinders and spheres.

B. Textures

Texture is an element of two-dimensional and three-dimensional designs and is distinguished by its perceived visual and physical properties.Use of texture, along with other elements of design, can convey a variety of messages and emotions.

OBJ is a geometry definition file format. The file format is open and has been adopted by other 3D graphics application vendors.

The OBJ file format is a simple data-format that represents 3D geometry alone namely, the position of each vertex, the UV position of each texture coordinate vertex, vertex normals, and the faces that make each polygon defined as a list of vertices, and texture vertices.

Textures in Unity is an image file, and it is used in different ways. The most common use of texture is when applied to a material in the base texture property to give a mesh a texture surface. Textures can be any image file supported by Unity, which means this can be photos from any digital source, but textures are usually images created or manipulated in image editor like Photoshop or Gimp.

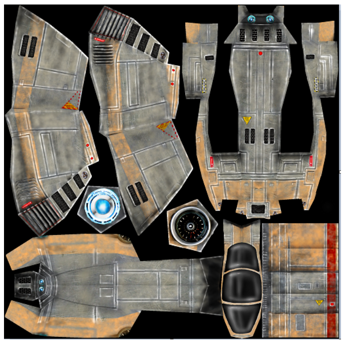

Fig. 1. Texture image of the Space Ship used in the game Project.

C. Lightning

Lights are an essential part of every scene. While meshes and textures define the shape and look of a scene, lights define the color and mood of your 3D environment.

Lighting in Unity is primarily provided by Light objects. There are also two other ways of creating light (ambient light and emissive materials), depending on the method of lighting you have chosen.

Illumination is determined by the interaction of the light source + the surface.

This section details the many different ways of creating light in Unity:

1) Point Lights: A point light is located at a point in space and sends light out in all directions equally. The direction of light hitting a surface is the line from the point of contact back to the center of the light object.

Point lights are useful for simulating lamps and other local sources of light in a scene. You can also use them to make a spark or explosion illuminate its surroundings in a convincing way.

2) Spot Lights: Like a point light, a spot light has a specified location and range over which the light falls off. However, the spot light is constrained to an angle, resulting in a cone-shaped region of illumination. The center of the cone points in the forward (Z) direction of the light object.

Spot lights are generally used for artificial light sources such as flashlights, car headlights and searchlights. With the direction controlled from a script or animation, a moving spot light will illuminate just a small area of the scene and create dramatic lighting effects.

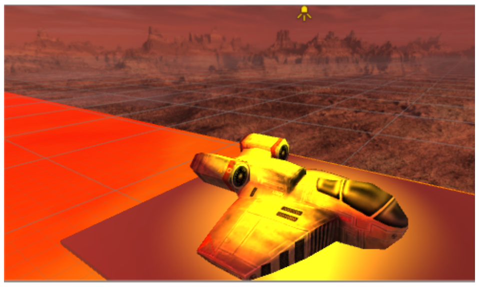

Fig. 2. Spotlight utilized in SpaceMan.

3) Directional Lights: Directional lights are very useful for creating effects such as sunlight in your scenes. Behaving in many ways like the sun, directional lights can be thought of as distant light sources which exist infinitely far away. All objects in the scene are illuminated as if the light is always from the same direction.

Directional lights represent large, distant sources that come from a position outside the range of the game world. In a realistic scene, they can be used to simulate the sun or moon.

4) Ambient Light: Ambient light is light that is present all around the scene and doesnt come from any specific source object. It can be an important contributor to the overall look and brightness of a scene.

Ambient light can be useful in a number of cases, depending upon your chosen art style. An example would be bright, cartoon-style rendering where dark shadows may be undesirable or where lighting is perhaps hand-painted into textures.

D. Camera

The camera is one of the most essentials components in Unity, it takes the content of our scene and displays it to the user. When a new scene is created one game object is always included, the Main Camera. The game view camera is a component attached to a game object, which means it can be manipulated or move our camera through parenting, scripting or physical interaction

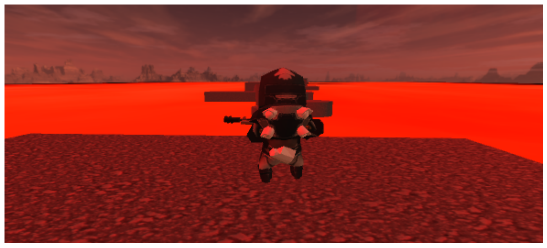

Fig. 3. Third person camera in the SpaceMan character.

To create a first or a third person camera, we can use the player object as the parent.

A screen space point is defined in pixels. The bottom-left of the screen is (0,0); the right-top is (pixelWidth , pixelHeight). The z position is in world units from the Camera. A world space point is defined in global coordinates (for example, Transform.position).

E. Collision Detection

Collision detection is the computational problem of detecting the intersection of two or more objects. To detect a collision in Unity requires one of the two colliding objects to contain a script with an OnCollisionEnter function in.

The fundamentals of handling a collision is summarized in 3 steps: collision detection, collision determination, collision response. In collision detection it checks if the object has collided with an other object, then in collision determination finds the intersection between objects and the collision response determines what actions should be taken in response to the collision of 2 or more objects.

F. SkyBox

For the background of the game i utilized a method called SkyBox which creates backgrounds to make a computer and video games level look bigger than it really is. Processing of 3D graphics is computationally expensive, especially in real-time games, and poses multiple limits. Levels have to be processed at tremendous speeds, making it difficult to render vast skyscapes in real-time.

Since the character is a space man, i downloaded a space skybox from Unity and applied to the scene [4].

III. Implementation

The implementation phase is essential for creating games, is basically the base that supports the game, the ”brain” of the operation. In game theory a implementation can be seen as a class of mechanisms whose outcome results in a given set of normative goals like making a character move or jump.

The game engine i utilized was Unity and the language for implementation was C#. Here is some implementations that i have realized:

A. Lava Flowing

In the implementation phase I started with the rivers of lava and how to make them move. I begin to attached a lava shader to a plane and then with a script, manipulated the textures creating some kind of movement to seem as it was really lava flowing [2].

When the character falls into the lava it resets the game all the way to the first position.

B. The character

To make the game complete it must to have a character, so i chose a space man on a mission to get to is space ship [4]. I downloaded the model from the Unity store, which come with animations and the textures of the character [1].

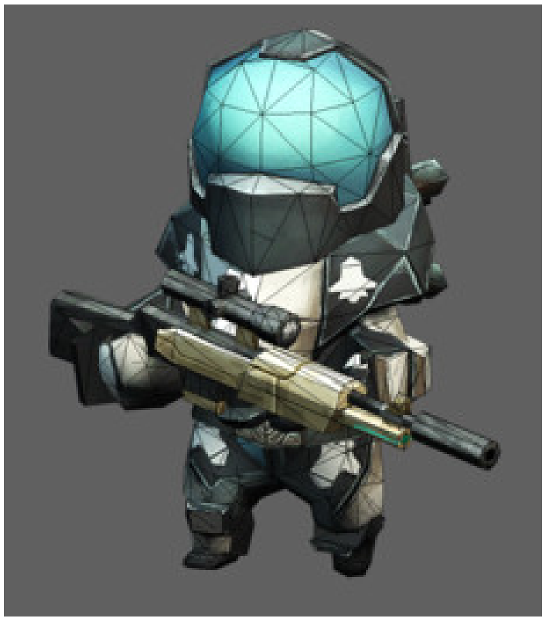

Fig. 4. The SpaceMan model.

1) Animation of the Character:

For the animation of the character i utilized the Animator Controller from Unity. Animator Controllers are state machines that determine which animations are currently being played and blends between animations seamlessly. Scripting is what allows us to really bring a model's animations to life.

The model had 2 animations states that i used, which was Idle (just standing still) and Run. To go from one to another state, it must to have transitions between them. The transitions are used in some conditions, like if i press the Vertical Key it goes to the Run state.

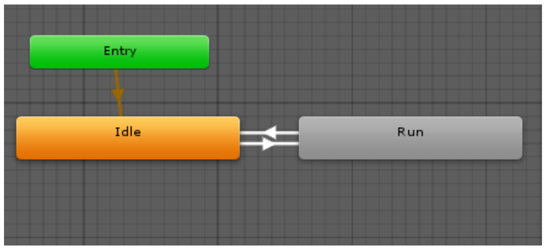

Fig. 5. Animation Controller states and their transitions.

C. Moving Platforms

For moving the platforms i define two positions where the platforms must start and finish, initial position and final position. Then i designate two arguments, smooth and reset time that control the speed of the fluency of the platforms and the time it takes to reset the movement.

D. Flickering of the SpotLight and ”You Win”

The main goal of the character is to get to the Space Ship [3] platform, so basically the user wins the game if achieves that final goal. For the flickering of the spotlight i used an IE numerator function with a timer to enable and disable the spotlight given some kind of flickering, so when the character touches the ”win platform” it makes the spotlight to flicker. For the sign ”You Win” similarly to the spotlight flickering when the character touches the platform i enable a canvas image.

E. Problems

The problems i majorly faced was in the implementation phase, specially in the animation of the character. Here's some of the problems i faced:

- The transitions of the animator controller was looping in same stages. In other words the character was animated always to run in a looping state.
- Some problems with the box collider of the character.

IV. Results

This project was very helpful in terms of bringing my creativity to a whole new level for me. I learned about the basic principles of computer graphics namely modeling in 3D, animation, camera control, texturing, illumination and rendering.

V. Conclusion

Summarizing, computer graphics is responsible for displaying art and image data effectively and meaningfully to the consumer, which means it's development has had a significant impact on many types of media and has revolutionized animation, movies, advertising, video games, and graphic design in general.

With this game project i conclude the demands that are required for a top level game to have, and how difficult it is to achieve that high level, but in a similar way i learned the bases that support computer graphics concluding that without those bases is not possible to make a high level game.

References

[1] Free Animated Space Man, HONETi Games.
[2] Lava Flowing Shader, MOONFLOWER CARNIVORE.
[3] Space Bomber, DUANE'S MIND.
[4] Skybox Series Free, AVIONX.