DRAGONFLY

COMP 371

Professor Adam Krzyzak

Team Members:

Nara Van Rossum

Sandy Khaled

Rich Park

TABLE OF CONTENTS:

Table of Contents

1. Goal

As a team, we had a lot of ideas for games and experiences for the user of our game. Though we had very divergent tastes, we found common ground from our enjoyment of several “experiential” games, primarily flOw and Flower. These experiential games are compelling primarily because of music and visuals. Working with the limitations of openGL and our learning objectives, our goal was create an interesting environment to explore while having the movement of our player-controlled agent affect the environment and sound.

The goal of this project was to essentially have an object being controlled by a player. Our controlled object was made to be a dragonfly, our dragonfly would initially be used to roam in this environment collecting smaller bugs as food and avoiding larger flies that would attempt to cross paths with it. Failing to avoid the larger flies would lead to a game loss. We wanted the visuals of our environment to appeal to the user so we really pursued the idea to provide a beautiful scene. This game was not intended to be a simple game where the user must collect all the smaller bugs to win; we heavily based it on the visual appeal where the user would have the benefit traveling in various themed environments. Each environment would have a different theme and atmosphere for the player to enjoy. Our concept challenges traditional gaming conventions since it has no defined objective, the goal is to journey around the world and jump into the different worlds merged in that environment. This game is targeted for individuals with an interest of being exposed to different visual effects and sounds.

2. Previous Work

Other people have tried to tackle the collision problem by making a collision cube around the object. The drawback to this however is that the more complex this object, the more collision boxes are required. This can result in collision boxes colliding with other collision boxes that are on the same object. This can produce negative consequences.

Irrklang library is used in many games to implement sound and music. One such game is World of Goo. IrrKlang is a library resource that is very easy to understand and does not cause many problems when working with OpenGL.

The particle system used to demonstrate water particles has been frequently used in the past to demonstrate reaction to water. They have been frequently used to portray fountains and waterfalls.

Creating a world by putting a sky texture on the inside of a sphere and placing a flat plane with a grass texture in the middle horizontally is a very common way to create the illusion of a type of "world". We will use this approach to create our main world for the dragonfly to explore.

3. Approach

We originally had quite a lot of ideas which were then condensed due to delays. We made a list of all the requirements that we felt that we could tackle. These were also mentioned in our proposal.

- Collision objects of various shapes
- Light sources
- Texture maps
- Camera to follow the user
- "Game Area" so user can't go off screen or go too far away from play field
- Models for Player, Targets, Enemies

We initially agreed to complete these points within a certain time frame. We attempted collision detection to do several of things, we wanted collision detection to prevent the dragonfly from going out of bounds and initiate a song corresponding to that sphere when the dragonfly enters it. The first few things that were completed were, texture mapping and light sources. The functionality of the camera movement was prolonged after it stopped working when all of our source codes were combined together. The dragonfly movement was successfully achieved but the camera movement that was applied to keep track of the dragonfly's projection path ultimately failed. Texture mapping was quite beneficial in our case since our game consisted to visually appeal the player. Since collision detection was scoped out, we also had to remove our wishes to create boundaries   which we had hoped to keep the dragonfly bounded to a region. By scoping out collision detection we shifted our focus in providing other features that would make up for the lack of collision.

Other approaches that we originally intended to implement were sounds and a particle system. The particle system worked well in our scene since we had water, creating a more relaxing environment.

4. Methodology

4.1 Implementations

To execute our approach we had to satisfy several requirements, these were: texture mapping, lighting, camera movement, object movement, object models. In order to create a visually appealing game our focus was primarily designated on having texture mapping and lighting applied correctly, music implemented to alternate the moods for each sphere and applying movement control of the dragonfly. We intended to a have a fun game rather than a challenging game. With that in mind we also implemented other special features in our scene, such as, fog and particle motion. Our particle system was used to create a fountain for our water in the main scene. This technique adds a little more emphasis to the movement of our water.

Our purpose for creating what we did is to provide the player we created a sphere with a moon texture on the outside, and the inside would be a blue sunny sky with a few clouds. A plane with a grass texture was placed in the middle to create a grassy area. This would create the illusion of a park or a field. We also created a river and a bridge over that river to give the world a bit of flair. Two fountains made out of particle systems were placed in the water to give it a little wow factor. Three smaller spheres were created, each with a different texture; a city, jungle, and space. A dragonfly the player controls will fly around the park and see that there are three different environments s/he can enter. When the dragonfly enters one of these worlds, a song plays and s/he can see different colored orbs flying around. The space sphere has bright orbs that float around really slowly, like a planet or a star. This gives the player a feeling of being in space. The jungle sphere has dark orange and black orbs flying very fast. This represents the stealth and quickness of predators in the jungle, and how fast one can be attacked by surprise. The city sphere has different colored orbs that spin around very fast in every direction. This sphere represents the business of a city and how crazy and hectic it can be. Each sphere also has a different song that adds to the vibe of each world.

4.1.2 Advantages/Disadvantages

In the end, we chose to simplify collision to be radius based; this has the advantage that it has a very simplified way to check what collisions might occur before the next render. This simplified method can check for both the inclusion and exclusion of a large number of objects quickly, but has difficult when dealing with irregular objects and when generating bounces off an angled surface. The disadvantages that we have faced during the course of our project are primarily due to collision detection. Collision detection was vital function that we eagerly hoped to implement but ultimately failed to do so. Instead we focused our attention to other requirements which were successfully met.

On the other-hand, given the methods for controlling a camera found through research, the idea we pursued, where the camera is more or less a collection of points upon which forces and translations could occur. This may have many benefits as a system for understanding and controlling a camera in a 3D environment. The disadvantages appear to be a reduced mobility of the camera’s point of focus and risk of unusual movement due to the combination of axis. The camera class may be solved better with use of quaternions; however the simplicity might be beneficial in certain circumstances.

4.1.3 Implementations Completed

- Camera with fluid motion and a fairly high degree of control
- Explorable world
    - world contains 3 sub worlds
        - each has a different texture, atmosphere, and song
        - each has different types of orbs of varying speeds which intend to represent different elements of the world
        - the program plays the appropriate song depending on which sphere currently contains the dragonfly
    - pretty fountain to watch
        - created by a particle system
    - bridge and river
        - gives the world a calming effect
- Animated and movable Dragonfly
    - Flapping wings and relative degree of motion and control
- Music that reacts to changes in the environment
    - Implemented using IrrKlang - A third party sound library

4.1.4 Implementations not completed

- Collision
    - Collision was very complex and we could not figure out a way to correctly implement a proper collision system that was effective given the other methods we were using
    - Some extended elements of collision handling were examined, such as collision prevention, and bounce methods

5. Results

5.1 Measure of Success

The way we have measured the level of success was based on how much was completed. We have completed about 70% of what we wanted to implement while the other 30% (which included collision detection and camera movement) was omitted due to their complexity and delays during the course of our project. Our concept gradually changed since we had to begin scoping out things that we had proposed on doing. We were ready to implement everything that we listed but our initial concerns were to implement the basic functions and then shift our focus on the additional functionalities. We were successful by means of creating an effectively appealing game that would drive the player to explore the various features that we implemented.

5.2 Experiments Executed

After learning about vectors, cameras and matrix revolutions in class, the ideas behind creating and controlling camera movement in a way similar to a 3d animation software really interested one of our group members. Nara focused on animation and bringing about smooth movements and controls in 3d. These experiments were often unsuccessful, but resulted in a final control rig for a camera that could circle and maneuver around an object of interest while easing through its movements. The task required a very attentive understanding of vectors and rotations in three dimensions. The problem that remained after the experiments was that of a camera following a moving target. This is an area that Nara might look more into, especially by examining some of the libraries for other programming languages which deal with cameras in three dimensions.

After extensive research, we were able to apply advanced techniques to visually compel the player to experience a game where the objective is based on exploration. Sandy was in charge of handling the main backdrop scene where the dragonfly would initially begin its path of exploration. We implemented three worlds which were contained within that environment. Rich created these three worlds by making three separate spheres, each with a theme very different from each other. These spheres each have a texture that represents the kind of world they are. One space, one jungle, and one city. These spheres would also rotate like a planet, to boost the illusion of a planet, or world. The space sphere contained smaller bright orbs that floated around, just like how planets would in space. This would give the player a convincing feeling that he is in a simulated space environment. The jungle sphere had dark orange and black orbs that would dash around the inside of the sphere. This simulates the stealthy and quick elements of predators in the jungle. The city sphere contained different colored orbs moving fast in random directions, to give off the vibe of a busy and hectic city.   Since those worlds had a very busy visual effect, we wanted to balance out the main scene with more tranquility. Sandy created a bridge by the use of polygons and cylinders and added texture mapping to create a wood effect. Moreover, she added water to create a more soothing effect to the scene, which was also texturized. To add more effect to the rather static water, she implemented a particle system to demonstrate fountains for the water. Keeping the scene simple she used a sky texture which had little texturing to balance out the already textured spherical worlds in the scene. She also implemented another effect to the scene that would change the mood of the scene to a darker theme. With fog implemented, the player has the ability to switch on the fog effect and fog color by a stroke of a key to perceive a more somber and darker environment.

5.3 Results

The result of our efforts is a small experiential interactivity with several unexpected elements and sounds which support artists working under Attribution and/or Share-and-Share-Alike licenses. The work will hopefully have some of its elements reused and become part of the greater creative commons.

6. Discussion

The approach we took was very promising because everything was planned out ahead of time, and we knew what he had to finish before we started something else. Naturally as we thought more and more, our idea evolved. If there was something we can change, it would be having more time to work on the project instead of assignments, and actually being taught how to implement certain functions that are required. A better approach would be to know what ideas would actually be possible earlier on, instead of weeks after the proposed idea.

We learned that anything that can go wrong will go wrong. OpenGL can be easy to learn at first, but the more that is added, the more complex the result becomes. We had difficulty finding modern sources of information, as often our research lead to websites that no longer exist, are no longer supported, or have outdated information. In the end, many of our team became disillusioned with what could be accomplished with OpenGL. As a group, we found that there are much better programs to model in 3d, and learning openGL helped our appreciation of those programs which can more quickly get to the creative elements of texturing, animating, higher level program design for implementing objects and game engines.

7. Conclusion

In conclusion, finding and implementing any method of complex effect takes a level of research more than what has only been known in the material that has been given to us. This has been learning experience that taught us that anything that can go wrong will go wrong.

Appendices

TEXTURES:

Space texture: http://allidoisthinkofyou.deviantart.com/art/L-I-A-P-texture-LARGE-174113905

City texture: www.prettylightsmusic.com

Water texture: http://www.shareaec.com/v/2987/gallery/6/Texture/Tileable-water-02

Sky texture: http://cgtextures.com/

Grass texture: http://loadpaper.com/id31614/green-grass-texture-01-by-goodtextures-on-deviantart-1024x681-pixel.html

Wood texture: http://www.hoskingindustries.com.au/blog/tag/wood/

Other textures:

Free open source

Music:

City World: Drift Away by Pretty Lights www.prettylightsmusic.com

Other songs are by M-PeX, and are under an attribution and share and share alike Creative-Commons license from FreeMusicArchive.com

USER MANUAL

Controls:

The camera is controlled by the **WASD** and **ZX** buttons. W tilts the camera towards the y-axis, S tilts the camera back towards the xz plane. A circles the camera to the left around its target point, while D circles the camera in the opposite direction. **Z** zooms the camera away from the target, **X** zooms the camera closer to the target.

The bug is controlled by **IJKL** and **UO** buttons. I increases altitude and speed of responsiveness to movement, while **K** reduces altitude and speed. **K** and **L** are orbiting around the center of the world, while **U** and **O** are forward and backward motions.

Fog is controlled through the **f** key, which toggles the effect. The **c** and **g** keys control the colour of the fog that appears. The **c** key allows for a cyan fog color while the **g** key allows a purple fog color.

In cases where the different music becomes difficult to trigger, the buttons **0**, **1**, **2**, and **3** can trigger the different songs.

Libraries and how to utilize them:

- IrrKlang library (for music)
    - place songs in project folder as .ogg files
    - place irrKlang.dll in the debug folder.
    - place irrKlang include folder in the project folder
    - place irrKlang lib folder in the project folder as well

```
- #include "include/irrKlang.h"
#if defined(WIN32)
#include <conio.h>
#else
#include "../common/conio.h"
#endif
#pragma comment(lib, "lib\\Win32-visualStudio\\irrKlang.lib")
using namespace irrklang;
```

  - put this code by the header calls

Hello! And welcome to the dragonfly tutorial.

To begin, we will present the following controls of our game.

Controls:

The camera is controlled by the **WASD** and **ZX** buttons. W tilts the camera towards the y-axis, S tilts the camera back towards the xz plane. A circles the camera to the left around its target point, while D circles the camera in the opposite direction. **Z** zooms the camera away from the target, **X** zooms the camera closer to the target.

The bug is controlled by **IJKL** and **UO** buttons. I increases altitude and speed of responsiveness to movement, while **K** reduces altitude and speed. **K** and **L** are orbiting around the center of the world, while **U** and **O** are forward and backward motions.

Fog is controlled through the **f** key, which toggles the effect. The **c** and **g** keys control the colour of the fog that appears. The **c** key allows for a cyan fog color while the **g** key allows a purple fog color.

In cases where the different music becomes difficult to trigger, the buttons **0**, **1**, **2**, and **3** can trigger the different songs.

Libraries and how to utilize them:

- IrrKlang library (for music)
- place songs in project folder as .ogg files
- place irrKlang.dll in the debug folder.
- place irrKlang include folder in the project folder
- place irrKlang lib folder in the project folder as well

```
#include "include/irrKlang.h"

        #if defined(WIN32)

        #include <conio.h>

        #else

        #include "../common/conio.h"

        #endif

        #pragma comment(lib, "lib\\Win32-visualStudio\\irrKlang.lib")

        using namespace irrklang;
```

- put this code by the header calls

After doing all this you are ready to play Dragonflyght, enjoy!