Advanced Games Technology Resit

Project Report

Daniel McKinley

**Summary:**

This document is the project report for my Advanced Games Technology resit coursework. It contains the details surrounding the project, including assets used and where they were found, features implemented with explanations, discussion based on the strengths and weaknesses of the game, along with what could be expanded on.

**Overview:**

The project in its current state has slightly more gameplay features than the first submission. The most notable changes overall are the fact that there is now a win condition (collecting all the gems) and a loose condition (getting to 0 health by walking into enemies). Additionally, I reworked the way physics are handled significantly, based on the game tutorials #37 project. The player can move around using WASD, jump using V, and shoot “grenade” projectiles by pressing Q. There are two types of collision detection; polygon based collision detection with the primitive shapes and floor plane (based on game tutorials #37), and AABB collision with the player, enemies (of which there are 5) and grenades that the player can shoot. I have imported a number of meshes which can be seen, and set up animation to work for the player’s mesh. The enemies work on a basic finite state machine, which has 5 implemented states: toofar, fight, flee, indifferent and dead. There is now the ability to do damage to enemies by shooting grenades at them, with a grenade SFX sprite when the grenade “explodes”.

There are three lights in the scene, the default light which was included in the template, a white point light that can be moved around using the num keys (1 - 6) which I used to check the vertex normals of the primitives, and a yellow colored light which is set above the LampPost primitive object.

Assets Used

## Downloaded:

* “Watch Tower Made Of Wood” 3d Model
  + <https://free3d.com/3d-model/watch-tower-made-of-wood-94934.html>
  + Downloaded before the first milestone date, around the 27th
  + Personal Use License
* “Muddy Soil Texture”
  + <https://jooinn.com/muddy-soil-texture.html>
  + Downloaded before the first milestone date, around the 27th
  + Non-Commercial license outlined here: <https://jooinn.com/s/license.html>
* “Old Gray Tin Metal Texture”
  + <http://www.photos-public-domain.com/2017/04/04/old-gray-tin-metal-texture/>
  + Downloaded before the first milestone date, around the 18th
  + Creative Commons CC0
* Player and enemy models are all from Gamers.org, which was a website supplied to us on Moodle
  + <http://www.gamers.org/pub/idgames2/quake2/graphics/md2/>
* “DST-BioIndustrial.mp3”
  + <http://www.nosoapradio.us/>
  + Creative Commons Attribution license
* The skybox texture was made using a WebGL app called Space 3D found here: <https://www.reddit.com/r/gamedev/comments/3ys4vg/a_month_ago_i_released_a_free_space_skybox/>

## Created:

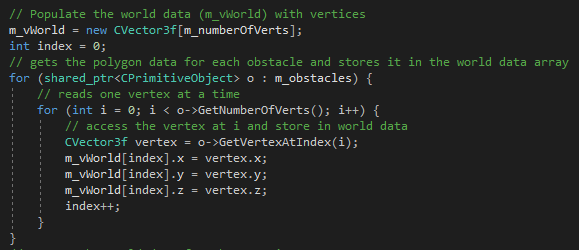
* I made the texture for the shipping containers and gems in Photoshop CC 2015.
* I made the splash screens in MS Paint

Part 1: Basic Game Modelling

## Intro screen with keyboard/mouse controls

To create the intro screen, I created a class called CSplashScreen. This class renders the splashScreen jpeg file using a full screen quad and Orthographic Projection mode. By using the m\_introScreen boolean, the player can stop the rendering of the intro screen by pressing a char. I added functions that change the image being rendered, which are called either when the player collects all the gems (win condition) or when the player dies by walking into the enemies (loss condition).

## Objects using OpenGL primitives; new skybox & terrain

For the OpenGL primitives, I’ve used lots of inheritance. I have 3 primitive based objects: Gem, MetalFloor, ShippingContainer and LampPosts, which all have their own class and header files. The rectangular ones are all “subclasses” of CPrimitiveCuboid, which is a subclass of CPrimitiveObject, which are both defined in PrimitiveObject .cpp and .h. I used inheritance since I knew that I wanted to use lots of cuboids for the sake of convenience, and so having lots of functionality in the base classes made doing things like collision detection or creating new instances of child classes easier. The gem on the other hand is a subclass of CPrimtiveOctahedron, which has a slightly different set of vertices.

I developed the collision detection from the previous game submission, based on the OpenGL game tutorials from GameDev.net (supplied on moodle) called “Sphere and Polygon Collision”. This uses the local coordinates of the shapes, transformed to their relative positions in the world coordinate system, to then calculate collision with the bounding boxes (the player and enemies). The major changes I made are that instead of each primitive object handling collision, that is all done by the player. The trick however is that all of the vertex data is read from each obstacle (pictured above) and constructs an array for the purposes of passing to the player and enemies so that they can collide with it all. The methods I used to do this are stored in the 3DMaths class and the Entity class.

For the ground texture, I found one online and changed the texture scaling in the terrain class so that it didn’t stretch one “square” of the texture over the whole terrain quad, but instead repeated it many more times. I decided to change the sky texture as I couldn’t figure out how to make the corners smooth, so I found a free tool online that could generate an actual spherical texture.

## Audio

For audio all I did was find a new piece of free music and use it in the place of the audio that was supplied in the template. I haven’t implemented any other sound effects.

## Heads Up Display

For the heads up display, I created a method called “SetUpUI” in the game class, which sets the member variables “health\_ui”, “shields\_ui” and “gems\_ui” to display the player’s health, shields and gems. You can see that these are dynamic by walking into an enemy or by picking up a gem.

Part 2: Camera, Meshes, Lighting, FX

## Camera Motion Technique

I followed one of the tutorials from the previous week quite closely (specifically the one for the third person camera), and building on that I created a strafe vector, which is then used to offset the camera to give the impression of an “over the shoulder” style camera. Another feature I’ve implemented here is the ability to flip which side the camera is on relative to the player, so that the player can peek either to the left or the right of a corner, instead of just being locked to having the player to the left of the center of the screen. This can be done by press “F”.

## Mesh Based Objects

There are 4 mesh based objects in my game. The most simple are the m\_watchTower and m\_mesh, which are simply set to load in models stored in the meshes folder, texture them and set and scale them into the scene.

The enemies each have their own model, which is simply set to their position and rotated to face the player in each frame. This model’s correct texture wouldn’t load properly, so I made a black .jpg in paint and used that as the texture instead.

The player, however, has a model which changes animation based on whether or not they are moving.

## SFX

I have implemented an explosion sprite class that is called by the grenades when they explode. Also, the default fog from the template is there.

Part 3: Physics, AI and Gameplay

## Physics

The player has physics acting on them, such as gravity, acceleration.

The grenades, which you can shoot by pressing “Q”, all have similar elements to the Football that was in one of the lab sessions, such as angular velocity, gravity and bounce.

## NPCs

There are 5 NPCs on the map, and they work on a basic finite state machine, which has 5 implemented states: toofar, fight, flee, indifferent and dead. If the player is too far away, they will go to the “toofar” start and pursue them until they get into range, at which point they will “fight”. The fight state is basically unimplemented, meaning they just stand still, but I would implement enemies shooting at the player if I had more time. Flee is basically entirely unimplemented, as I would have used this by having the enemies enter this state if they got too low on health, but I didn’t get around to this.

## Power Ups

The gems are pick ups, and collecting all of them make the victory screen pop up.

Part 4: Discussion

If I were to develop the game further, I would use a BSP or Octree to subdivide the level geometry data to reduce the compute power needed to perform the collision detection, especially considering I would have a more developed map if that were the case. I would also probably "bake" the static map elements in the game that are children of the PrimitiveObject class like the shipping containers into a seperate map file, as is done in Game Tutorial #37.

Enemy and gem spawn locations are currently random, and set in the constructor of the Enemy class. This means that these can sometimes spawn inside of other objects, or even potentially on top of the player, for example, which would naturally be undesireable in a finished product as this would only frustrate or confuse the player. To fix this, I think it would be best to add a "SpawnPoint" class which can be stored in two shared\_ptr vectors in the game class and interated through to initialise the enemy and gem positions in preset locations.

Finally, the game class is getting very bloated, and has low cohesion and high coupling which is undesirable. If I were to develop the game further, I would probably refactor the game class in its entirety and add some helpful engine features like a messaging system or events manager to help reduce the complexity of the game for the sake of myself and other programmers sanity. The code in general is quite badly organised, so if I were to come back to the project I would refactor is massively.

# Known Bugs:

* Though I've reworked the way collision and collision detection is done in the game, there is still a strange issue where if an entity tries to walk into a corner of one of the obstacles (shipping containers, etc) the offset that the entity is set to seems quite buggy and won't let them "slide" along the surface as they do when walking into a flat surface.
* While working on this project, my desktop PC had a major malfunction and was out of commission for a day, so I had to work on my laptop for a bit. I noticed that the game's physics were significantly different on the laptop (even though it was running the same game version), i.e while jumping on my laptop the player barely got off the ground, whereas on my desktop was functioning as intended, with a much higher jump. To get the jumping as it was intended on my laptop, I changed the scaling applied to the m\_instantaneousAcceleration in the Jump method on the Player class. Please keep this issue in mind if the jumping physics seem too underwhelming/overwhelming.