**Team**Cool Bananas

*Assignment Two:*

***3D Game (Gold Master)***



**Team Members:**

**11688025** Deinyon Davies

**11685740** Daniel Nelson

**11697822** Matthew Wale

**11729631** Jesse Walker

**Introduction**

Welcome to **Teapot Defence Simulator 2014 (TDS2014)**! TDS2014 is a fantasy-themed tower-defence styled game in which the player must stop enemies from destroying their home base. Waves of enemies spawn in and make their way to the player’s base, finding the most efficient path to get there. To find the most efficient path, these enemies navigate around obstacles, and have tendencies to stick to specific ground types that allow faster travel.

**Controls**

|  |  |
| --- | --- |
| **Operation** | **Action** |
| Mouse Movement | *In FPS and Flying modes:* Rotate the camera |
| Left Mouse | Fire a projectile **or** place a Wall Tower in the game world |
| Right Mouse | Pathfind to the selected block in the Pathfinding Sandbox |
| Middle Mouse | *In Top-Down mode*: Press and drag to rotate the camera |
|  |  |
| W,A,S,D Keys | Translate the camera about the world |
| Spacebar | *Only in FPS Camera mode:* Have the player jump |
| Shift | Increate the translation speed of the camera or player |
|  |  |
| 1 Key | Use an FPS Camera |
| 2 Key | Use a Top-Down Camera |
| 3 Key | Use a Flying Camera |

**How to Play**

In the setup phase, the player uses currency to purchase and place towers (left mouse button) and obstacles on valid ground squares of the map. Once the player has set up their defences, they can enter the defence phase of the game, where wave of enemies begin to approach the player’s home base. The towers that have been placed will obstruct the paths of the enemies. Enemies will navigate around these towers while being attacked by the towers.

**Technical Features**

The game exhibits a three-dimensional world composed of multiple static models, non-static entities, and NPC instances. The player can navigate the world using the keyboard and mouse to control the camera, and certain objects within the world can move from point to point, and stop this movement upon collision. Players interact primarily via mouse and keyboard input for tower placement and camera movement.

Players can score points and currency by successfully defeating the waves of enemies that are working their way to the player’s base. Players either succeed in destroying all enemies, or fail upon destruction of the base.

The project exhibits **34** distinct classes (…number has likely changed), of which some contain nested classes. A great deal of polymorphism has been employed to reduce code repetition and to greatly enhance maintainability. Development of new components is made easy by the high level of abstraction and comprehensive framework.

The player, cameras and NPC entities observe the laws of game physics through the implementation of Steering states and Kinematic physics objects. In most cases, entities within the game are translated by means of physically accurate steering forces and velocity computations.

NPCs within the demonstration employ a basic state machine and a *priority stack* to select the most appropriate steering behavior for the situation.

**Four** steering behaviors have been implemented, of which **three** are demonstrated by the two enemy tank NPCs through polymorphic steering classes. The tank usually nearest to the player will **pursue** the player with consideration to their velocity and position. The other tank will **arrive at** the pursuing NPC. Both tanks will **avoid** geometry using a collision-avoidance algorithm.

(Honestly not too sure about this one) Spatial partitioning has been implemented to some extent and does… something.

An advanced **A\* Pathfinding Algorithm** has been implemented, and is demonstrated in the waves of enemies that navigate through the level. The algorithm uses weighted search nodes to traverse the terrain using the fastest and most logical route. It is possible to re-form the search nodes by placing new Wall Towers with the left mouse button.

The program plays multiple **sound effects** and an **ambience loop** using a dedicated Sound Manager class which is responsible for loading and maintaining a list of sound effects.

Several XML configuration files are now integrated with the game. These configuration files allow the player to easily find and edit XML tags that will modify the structure of the map, as well as key game parameters. The structure can be edited as a grid representation of the world. Different characters will represent various types of tiles, and the player can easily open the XML document and change the level at will. Key game parameters that can be modified include wave properties, player health, tower properties, and so on. Difficulty can be changed based on the values edited into the configuration files.

There are also XML files which allow for basic AI scripting. Much like the other configuration files, the user can set parameters and conditions for the NPC behaviour. For example, they can set the size of the radius in which the NPC switches from the idle state to pursue, and so forth.

A custom written HLSL shader program has been composed that generates a plasmatic per-pixel effect with the trigonometric functions, and projects the effect onto a custom model using its texture coordinates.

(This will need to be modified based on the current collision stuff… which I should probably know about) The game demonstrates **two** collision response algorithms. Projectiles fired from the enemy tanks or the player, performed by the ProjectileManager, are tested for intersection against the world and the player - when an intersection occurs, a sound plays, and in the case that it intersects the player, damage is taken and the camera shakes using a Thread delegate function. Enemy tanks implement a ray-sphere intersection algorithm to compute an opposing steering force, and steer away from the nearest collision threat. The nearest collision threat is found using the ray-sphere intersection algorithm, which returns the amount that the NPC's 'eyesight' ray has intersected with the geometry.

The game demonstrates a number of custom models, a custom HLSL plasma shader, a projectile management system, a sound management system, and a weighted A\* pathfinding algorithm.

**Peer Evaluation more like this will be left for later evaluation ha ha yeah**

*Deinyon*: Lead Programmer, Graphics & Documentation Expansion

**Score:** 5.0

Contributed the majority of the codebase and framework, implemented collision response algorithms, designed graphics and contributed to the readme document.

*Daniel*: Designer & AI Programmer

**Score:** 5.0

Researched A\* path-finding and implemented a demonstration for it. Also led design talks and contributed lab code to the project.

*Matthew*: Design, Documentation & Administration

**Score:** 3.0

Organised, wrote, and compiled several elements of the read-me document. Recorded and edited the video for the assignment, along with uploading to YouTube. Participated in group design meetings.

*Jesse* : Documentation & Designer

**Score:** 3.0

Participated and helped with group discussion, contributed lab code to the project, and helped with elements of the read-me document.

**References**

***Tutorial & Reference:***

Amit's A\* Pages 2010, *Pathfinding,* viewed 3 September 2014,   
<http://theory.stanford.edu/~amitp/GameProgramming/index.html>.

Gamedev Tuts+ 2013, *Understanding Steering Behaviors : Collision Avoidance*, viewed 4 September 2014,

<http://gamedevelopment.tutsplus.com/tutorials/understanding-steering-behaviors-collision-avoidance--gamedev-7777>.

Patrick Lester 2005, *A\* Pathfinding for Beginners*, viewed ~25 August, 2014,  
<http://www.policyalmanac.org/games/aStarTutorial.htm>.

***Textures:***

Jacobo Cortés Ferreira, *Grass0139\_33* (resampled), viewed 3 September 2014,  
<http://www.cgtextures.com/texview.php?id=44371&PHPSESSID=pegmiou3bj4bkrbgvc54jvna25>.

CGTextures, BrickOldRounded0298 (Resampled), viewed 3 September 2014,  
<http://www.cgtextures.com/texview.php?id=52593&PHPSESSID=j6fcl84hlqk61c723qjsdstsq4>.

CGTextures, SoilCracked0157 (Resampled), viewed 13 September 2014,  
<http://www.cgtextures.com/texview.php?id=66844&PHPSESSID=eimacm8d4ftmecu9p3642eo1b4>.

R. Boel, 2006, *Sun altitude of 5 degrees above the horizon*, Roel z'n Boel, viewed April 19 2014, <http://reije081.home.xs4all.nl/skyboxes/>.

Pokemopolis Forum, *Awesome Smiley*, viewed 13 July 2010,  
<http://captionsearch.com/pix/thumb/nqfzkzcwdr-t.png>.

***Audio:***

Valve Corporation 2004, *metal\_barrel\_impact\_2*, sound effect, Half Life 2 Resource.

Valve Corporation 2004, *shotgun\_fire6*, sound effect, Half Life 2 Resource.

Valve Corporation 2004, *smg1\_fire1*, sound effect, Half Life 2 Resource.

Valve Corporation 2004, *towm\_ambience*, sound effect, Half Life 2 Resource.