**Project Overview**

**Team Number : 4**

**Team Members :**

* **160540A / Lai Wei Jing, Eugene**
* **163940Z / Chan Zhihao**
* **161725E / Lim Guan Sheng, Marcus**
* **160853T / Kwong Yu Ming**

**Game Concept**

The player is has to kill enemies and destroy towers in order to proceed on with the game.

**Technical Goals**

A-Level:

* User-Friendly interfaces
* Simple AI
* Basic Particle systems

B-Level:

* Optimized Particle systems
* Believable AI

C-Level:

* Shadows

**System Requirements**

**OS :** Windows 10

**Technical Risks**

* Amount of enemies on the map can cause lag on low specification computers.
* Enemies might not be able to trace player properly upon hitting a wall.

**Third Party Tools**

**irrKlang**

irrKlang is a high level 2D and 3D cross platform (Windows, Mac OS X, Linux) sound engine and audio library which plays [WAV, MP3, OGG, FLAC, MOD, XM, IT, S3M and more file formats](http://www.ambiera.com/irrklang/features.html#formats), and is usable in C++ and all .NET languages (C#, VisualBasic.NET, etc). It has all the features known from low level audio libraries as well as lots of useful features like a sophisticated streaming engine, extendable audio reading, single and multithreading modes, 3d audio emulation for low end hardware, a plugin system, multiple rolloff models and more. All this can be accessed via an extremely simple API.

**Game Play**

**Game structures/ Game objects**

The following is a list of logical objects currently planned for the game:

* Wall - A wall
* Sword - Melee weapon
* Gun - Mid to Long range attacks
* Crate - Low obstacle for player to take cover
* Towers - To be destroyed by player

**Characters:**

Main character

Robot

Enemies - Protecting the tower (attack player)

NPCs - Allows player to buy stuff(weapons, items)

**Physics**

Based on a simplistic Newtonian model

**Newtonian**

A simple velocity and acceleration based physics model that handles gravity and simple collisions. This is used by objects to modify their position based on the internally managed velocity. It contains: - The mass (kg) of the object. This is used for collisions. - The ratio of energy it loses on impact. So 0.1 is very rubbery and 0.9 is very sticky. Negative values will gain energy on impact. - Current velocity (m/s). This is not normalized as it also has a magnitude. - Speed that is automatically calculated as the magnitude of velocity. - Gravity acceleration that is applied every frame. By default it's 9.8 (m/s^2). - Accumulated acceleration is used for efficient processing of outside forces. Any calls to accelerate a Newtonian are summed in this variable and are used during normal updates once per frame. - Number of collisions is the total number of collisions that have been reported and represents it as only one collision that frame. - A ground Boolean that keeps track of if the Newtonian is resting on a surface. This activates a slightly different model that ignores gravity and uses friction. An “Accelerate” function allows outside forces to act on the Newtonian. Newtonians can also be accelerated by using helper functions that handle physical collisions between each other and with a static plane. When all accelerations are processed the velocity is capped at the maximum value 80m/s.

**Projectiles**

**Collision**

**Entity Manager**

Handles collision of all entities. Functions will be made to check collision between objects. This is to prevent any overlapping of objects. Collision will also be used for hit detection.

The collision logic used are:

* Axis Aligned Boundary Box
* Ray-Sphere Collision Detection

**Player actions**

* Walking around (First Person View)
* Using weapon

**Victory Condition**

* Down the towers.
* Defeat the officers stated.

**Level Specific Codes**

Tutorial:

Specific code required:

* Holding keys for a certain timeframe before proceeding.

**Artificial Intelligence**

Artificial Intelligence for the game will be broken up into various state machines for different game objects (enemy).

**AI Structure**

Every object that is said to have AI, will have a set of C++ files (.h and .cpp) associated with it that will be the character’s AI manager (CEnemy3D). For instance, Troop will have files called Troop.cpp and Troop.h. These files should contain a class that has general information about the character, such as AABB boundary, attributes (Health, Attack and Defense) and state machines.

**Code Overview**

**File formats**

All code should be contained in standard C++ source files (.cpp) and standard C/C++ headers (.h). The project workspace will be contained within a Microsoft Visual Studio .NET solution (.sln). Levels will be contained within a comma-separated values (CSV) file.

**Comments**

Single line comments should be used as often as humanly possible to elaborate code flow. Specific marker tags: "TODO:" should be left in places where the coder is skipping small functionality for the time being to get a major feature done first. TODO's should elaborate what features need to be implemented later. "RECODE:" should be left in places where the coder knows they're being sloppy. Recodes should contain suggestions for better code if ideas are available, but at the very minimum should point out specifically what is sloppy.

**Source Control**

All code modules will be backed up in our shared Google Drive folder. Version control is based on GitHub and SourceTree application.

**User Interface**

**Game Menus**

**Startup Menu:**

The startup menu will consist of a 2D menu with the following choices:

* “Play Game” - this opens the Game Mode Menu
* “Options” - this opens the Options Menu (change controls and sound level)
* “Exit” - exits program

**In-Game Controls**

*Controls can be changed in the options*

**Movement:**

* W/A/S/D - movement
* Space - jump
* Shift - Sprint
* Ctrl - crouch
* I - access inventory
* Left click - attack (use weapon/ fires weapon)

**Looking Around:**

Designed to use mouse to free-look, but keyboard will be provided for players who need it.

**In-Game Menu:**

The in-game menu can be brought up with a command that will be default bound to the escape key. The menu will pause the game and bring up the following options:

* "Return to Game" – Un-pause the game and continue playing.
* "Restart Room" – Restarts the current level. A confirm screen may be used depending on focus group results.
* “Options” – Loads up the same options menu specified in the “Game Menus” section above. The player can change video/audio settings and rebind their controls.
* "Quit to Main Menu" – Takes the player back to the main menu. A confirm screen will definitely appear.
* "Quit Game" – Quits back to Windows desktop. A confirm screen will definitely appear.

**Death Menu:**

Upon death, the player will be respawned at the town.

**Win Screen:**

When the player conquers all the towers, something will happen to acknowledge this triumph.

**Graphics**

OpenGL will be our choice for graphic engine.

**Features**

**Formats**

**Textures**

.tga files will be used for texture files.

**View Modes**

2 different viewing modes (third person mode, if time permits)

1. First person mode
2. Top-Down 2D mode
3. Third person mode (Extra task)

**Task List**

**Eugene:**

* Collision detection
  + Ray-sphere collision detection for gun based equipment.
  + Prevent camera from moving if next position is an object.
* Map Editor
  + Placing objects and enemies on the scene for easy level creation.
* A.I
* Level 4

**Lim Guan Sheng, Marcus:**

* Inventory
  + Switching of different equipments.
* Level 1

**Yu Ming:**

* Player Attributes
  + Statistics such as Health/Magic/ power , Strength and Defense
* Level 2

**Zhihao:**

* Cinematics camera system
  + Moving camera according to waypoint
* Level 2

**List of tasks**

**Camera**

* First Person View
* Top Down View
* Third Person View

**Player**

* Move forward/backward/left/right
* Jump
* Crouch
* Sprint
* Stamina
* Collision

**Mouse/keyboard controller**

* Mouse Controller
  + Left click (Attack / Use / Clickable item)
  + Mouse movement (View around)
* Keyboard Controller
  + W / A / S / D ( Player movement )
  + Left Shift (Sprint)
  + Spacebar (Jump)
  + I (Inventory)
  + P (Pause)

**AI**

* Enemy
  + States
  + Collision
  + Health
* Tower
  + States
  + Health

**Inventory**

* Weapons (Melee / Gun)

**Power-ups**

* Health / Stamina pack
* Weapons

**User Interface**

* Main menu
  + Level Selection?
* Gameplay Scene
  + Inventory
  + Minimap
  + Health / Stamina bar
  + K.O Counter
* Pause menu
  + Return to game
  + Options
  + Quit
* Win / Lose Scene

**Sound**

* Sound Effects
* Music

**Timeline / Milestone**

**Week 1 Schedule :**

* Basic mechanics
  + Proper ray-sphere collision detection when projectile fired
  + Switching of equipment via inventory
  + Transition between scene to another
  + Camera panning to various location before level gameplay starts
  + Damage system based on attack, defense and health power

**Week 2 Schedule:**

* Add-on features
  + Save/Load Game
* Debugging
* Playtest

**Week 3 Schedule:**

* Polishing features
* Ensure game is running in proper condition(No crashes/Reduce bugs)
* Documentation of game

**Installer**

Installer will be made using the iexpress.exe. <http://www.wikihow.com/Make-an-Installation-File>