DM2294 Game Development Project

Technical Design Document

Team E

**Team members:**

**144112K Edmund Ang De Xiang (Leader)**

**140522J Almeda Glenn Paolo Maestro**

**141947W Wee Jun Yang Gerald**

**143182M Toh Yi Fun Troy**

# Game Concept:

The player is a prisoner whose been caught stealing some fine literature in a store and have been sentenced to three months of jail term. Your friend tells you that there is a performance by Mr. Toh Da Jun and he will be featuring your favourite “Men By The Beach” song. You want to join him and so you decide to break out of prison.

# Technical Goals:

* 3D Graphics
* 3D sounds
* Simple AI with pathfinding
* Particle system
* AABB Collision Detection
* Collision Response
* Modular development
* Shadows
* Dynamic Lighting
* Dynamic weather
* Item interaction
* Frustum Culling

GAMEPLAY

# Game structures/ Game Objects

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

* Wall – Basic wall
* Key card – To obtain and unlock doors
* Doors – Security doors to be unlocked
* Night vision googles – A power-up to see in the dark better
* Invisibility cloak – A power-up to avoid detection
* Laser fence – A inferred laser to detect player while invisible
* Table – Props to the world

## Characters:

* Warden – The person in charge of the prison
* Guards – The people patrolling and wandering around the prison
* Inmates – The people who are in the same position as the player

# Physics:

Our physics would be a simple simulation of gravity, common wind direction and AABB collision detection. Our engine will control the time, lights and the weather.

* Gravity – 9.8m/s² downward force.
* Wind direction – Every 30second in real life changes the Vector3 direction, particles are affect by this.
* Time – 1 second in real life is 1minute in the game, it takes 25mins in real world to reach full 24hours in the game.
* Light – Light will change and simulate the sun effects
* Weather – Changes accordingly to the time base in the game and activation of the rain doll

## Collision:

### AABB Collision detection

Each object has 8 points aligned to the 3 axis of the world and it checks against the player’s hitbox of 8 points. It returns a collision response if detection is true.

# Player actions:

The player are capable of the following actions:

* Sprinting
* Walking
* Jumping
* Crouching
* Urinating
* Picking up items
* Use power-ups
* Using key cards to open doors

# Victory conditions

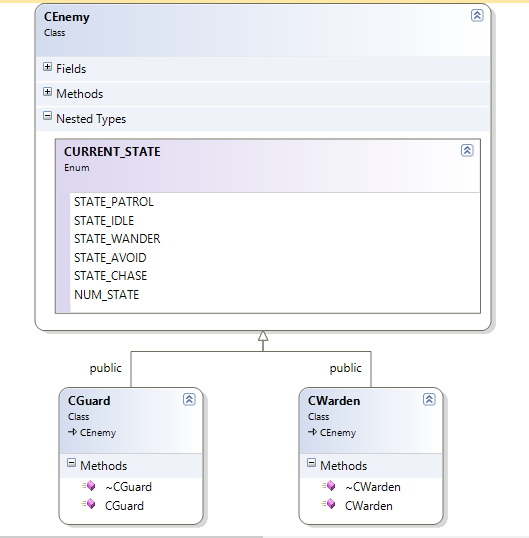
The player completes the level when a certain objective is achieved in each level

* Level 1
  + To collect the yellow key card and open the yellow door
* Level 2
  + To steal the red key card and open the red door
* Level 3
  + To activate the red button to release prison locks
* Level 4
  + To reach a certain zone in the level to escape

Artificial Intelligence

The AI for the game will inherit a common CEnemy.cpp with a state and have their own .CPP and .H file. The AIs will be based on 3 main states.

# AI Base States

* Idle – The state where the AI does nothing but still keeps a look out for the player
* Wander – The state where the AI randomly wanders around the level and keeps a look out for the player
* Chase – The state where the AI will find the path to the player and chase him until the player is caught

Code Overview

# File Format

Codes should be contained in their standard C++ source files (.cpp) and standard C/C++ headers (.h) respectively.

Variables are to be loaded by a text file (.txt) and AI pathfinding to be loaded by comma separated values (.csv).

# Comments

Single line comments should be used as often as possible to elaborate code flow.

# Naming convention

Multiple word names should have an underscore to separate the words or a capital letter of the starting of the next word.

Variables should follow the standard Hungarian notation where applicable.

Accessing private variables of a class should have their respective Get/Set functions implemented in each class for each variable.

# Source Control

All codes are to be kept and updated by using a central repository in GitHub. Members are to ensure that codes being pushed into the repository should compile with errors unless stated otherwise. Repository should ignore junk files that can be obtained by compiling.

User Interface

# Game Menus

## Startup Menu:

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

* “Play” – To enter the first level and play the game
* “Instructions” – To display the controls and description of the game
* “Credits” – To display the names of the people working on the game
* “Exit” – To end the application and return to Window’s desktop

## Pause Menu:

The pause menu will the player’s in-game screen with a translucent screen overlay with the following choices:

* “Resume” – to un-pause the game
* “Restart” – to restart the current level
* “Exit” – to exit back to main menu

# Game User Interface



1. Player’s current keycards
2. Compass that points North
3. Enemy detection warning
4. Watch to display current in-game time

In Game Controls

* Movement
  + Movement will take the standard 4-command strafing setup of W/A/S/D.
  + Crouch by pressing “Ctrl”.
  + Jump by pressing “Spacebar”.
* Looking around – The camera will be controlled by the mouse
* Pause menu – The pause menu can be brought up by hitting the “ESC” key
* Night vision googles – NVG can be activated by press “Q” once obtained
* Interaction – The general interaction button is “F”

Graphics

OpenGL 3.3 will be used in this development throughout the entire duration as we are more familiar with it compared to other rendering methods.

# Formats:

Textures are to be done in Targa files (.tga) either in 24-bit or 32-bit.

Object meshes are to be saved in object files (.obj) and generally should not exceed 1,000 vertices.

Heightmaps for terrain are to be saved in .RAW file formats.

# View mode

The game is to be played in First-Person view.

Audio

The audio engine to be use is irrKlang 1.5.0 as it is free for non-commercial use.

# Features

* 2D/3D sound effects
* 2D music supported
* Allows music/sound stacking
* Free to control music volume

# Formats

irrKlang supports many different audio format but this application is to use only the following format for consistency

* MP3
* OGG
* WAV

Task List

Edmund

* AI classes and algorithm
  + AI automatic waypoint generation
  + AI State
  + AI Field of Vision
  + AI state machine
  + AI pathfinding
* Framework
  + Frustum Culling
  + Provide framework
  + Physics Engine
    - Dynamic Light
    - In-game time system
    - Collision Detection
    - Collision Response
  + Load from text file
  + Fix warnings
* Base Classes for teammates
  + Player class
  + Enemy class
  + Object class
  + Item class
* Objective Markers
* Compass
* Enemy warning detection
* Level 2 design

Glenn

* Blah

Gerald

* Blah

Troy

* Blah