### Technical Design Document Template

1.0 Revision History

<As you revise the document, list what was changed and when it was changed>

|  |  |
| --- | --- |
| Version | Description |
| **1.0** | **Initial document** |
| **1.1** | **more description of AI path finding** |
| **1.2** | **improved detail of flow chart.** |
| **1.3** | **edited revision history and mentioned 4 direction movement** |
| **1.4** | **Added descriptions and discussion of third party graphical framework.** |

2.0 Development Environment

2.1 Game Engine

<Proprietary/Unreal/Unity and version>

**Proprietary**

2.2 IDE

**Visual Studio**

2.3 Source Control procedures

**Github Repository (**[**https://github.com/rabbet35**](https://github.com/rabbet35) **Repo named AIForGames)**

2.4 Third Party Libraries

**RayLib, RayGui**

**Description:**

**Suitability for this application: Very good, simple implementation and light weight.**

**Technical impact: Minimal, light weight framework and easy to implement.**

**Licensing: No need to pay royalties and permission is granted to modify the library to the specific needs of this application.**

2.5 Other Software

<2d art assets, audio, 3d modelling etc.>

**None**

3.0 Game Overview

3.1 Technical Goals

<3d graphics, 60fps, Challenging AI etc.>

**Semi-realistic guards that respond to random stimulus**

3.2 Game Objects and Logic

<A list of logical elements in the game, i.e. door, button, pistol, ammo, light, bullet, wall, character etc. and description of their behaviour and purpose>

**Player Entity**

**Guards**

**Start/Exit location**

**Goal (Object to steal)**

**Walls (Maze like level)**

**Navigation Nodes for Guards**

3.3 Game Flow

<description of what the player can do (actions) from the start menu to playing the game, through to hitting quit. Include how to win, how to lose, how the player is moved, and what programmer things might need to be considered>

**Player can input movements using arrow keys. Player starts at starting position. Player must get to the object to steal without being caught and killed by guards. Guards kill the player when they are touched by them. Player must retrieve the object and return back to the starting position to win.**

4.0 Mechanics

<A list of the core game mechanics. I.e., what the player can do and how they achieve this, and what this triggers in the game. For example, shooting enemies is a core mechanic in an FPS>

**Player can input movements using arrow keys. Player starts at starting position. Player must get to the object to steal without being caught and killed by guards. If Player enters guards line of sight then all guards start to seek the player, and kill the player if they touch the player. This means the player loses. Player must retrieve the object and return back to the starting position to win. When the player wins, the win score increases and the game resets. When the player loses, the loss score increases and the game resets.**

5.0 Graphics

<Describe graphics features here. I.e., is your game top-down 2D? >

**Top-down 2D, Primitive shapes and colours to represent all game objects.**

6.0 Artificial Intelligence

<Describe how AI works, i.e. state machine, fuzzy logic, GOAP. Describe the various behaviours and how they change behaviour, how do the ‘creatures’ in the game evaluate the world> <include diagrams/flowcharts showing decision making processes>

**The guards will use the navigation nodes of the level to find the closest path to the player when seeking them. Or, when losing line of sight, will navigate to the last node the player was seen at. The method of path finding will be basic Dijkstras.**

**The Navigation nodes will be placed everywhere where there isn’t walls. Nodes links will have a weight of 1. Guards will travel to their next node by simply aiming at it and walking forwards. The node graph will allow for 4 direction movement.**

**Guards AI will be a state machine.**

**The state machine will work as follows:**

**STATES:**

**WONDERING AND SEEKING**

State = WONDERING

NO

NO

YES

State = SEEKING

7.0 Physics

<What engine are you using, what features from it (spring? Colliders?) how will physics be handled for objects? (box or sphere collider for objects, capsule for player) need to record specific locations for any reason? Potential slowdowns and how to mitigate.>

**Basic Axis Aligned Bounding Boxes for player, guards, walls and triggers. Basic collision resolution, ie, no bouncing or friction, just clipping.**

8.0 Items

<List of items you can pick up that can affect the player, and what they will affect, like ‘picking up the hammer (refer collisions above) adds 5 to the players attack attribute’. Include details on how items influence gameplay or AI logic.>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item | Parameter | Parameter | Parameter | Description |
| **Object to steal** |  |  |  | Must be obtained before player can win |

9.0 Game Flow

9.1 ‘Mission’ / ‘Level’ structure

<Are all levels stored in memory? what data is saved across levels, are levels loaded synchronously to prevent pauses?>

**There is one level. It is stored in code.**

9.2 Objectives

<What does the player try to accomplish on each level/mission? How is the players progress evaluated?>

**The player must obtain the object to steal without being killed and return to the starting position.**

10.0 Levels

<If any of the Levels require specific behaviours, describe those here>

**The main level requires that the guards use navigation meshes and line of sight to try to catch the player. If the player goes around a corner, the guard will go to the last spot it saw the player.**

11.0 Interface

11.1 Menu

<What are the menu options and what do they do?>

**No menu or options.**

11.2 Camera

<Describe the camera, how it moves, perspective/orthographic, can it switch? How? Does it need to render-to-texture? does it prevent itself going through walls, use flowcharts to document behaviour>

**There is a single top-down static camera which does not move, collide or change properties.**

11.3 Controls

<Keyboard, tablet touch/swipe/tilt, joystick, mouse etc. record double taps, multi touch, use mouse smoothing/ scale mouse for aiming etc.>

**Keyboard input, arrow keys used to move player.**

14.0 Asset List

<List all files needed, along with known attributes >

**Standalone executable**

16.0 Technical Risks

<if you want your game to be a 1000 player pvp battle royale with 4k 120fps graphics, you need to say if this is doable and how you intend to do it>

**No technical risks apart from developer competency.**