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 |
| 2.0 | Updated document |

2.0 Development Environment

2.1 Game Engine

<Proprietary/Unreal/Unity and version>

The game will be running on visual studios with Raylib in use.

2.2 IDE

Using visual studios for this project.

2.3 Source Control procedures

1) set up repository.

2) every 1hr - 2hr, save and commit work.

3) check to see if save was successful

2.4 Third Party Libraries

Raylib libraries (raylib.lib).

2.5 Other Software

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

Kenny - free asset store (2D models)

3.0 Game Overview

3.1 Technical Goals

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

2D top-down stealth/survival game with a simple AI that can ATTACK, CHASE the player, if it can't find the player, it will WANDER the map.

3.2 Game Objects and Logic

The game logic will be simple to start with and then expanded upon, known game objects include:

- **wall** = will repel any (game)object that collides with it, AI cannot move through this game object.

- **health station** = will restore/heal health to the player by a random amount (1hp - 10hp / 100hp, per pack). AI cannot move through this game object.

- **hp packs (counter)** = this will be a count of how many "packs" are available at the time and will affect the state-machine if its count reaches zero, the health station will be unavailable.

- **player (user)** = this will be a class/game object that will be controlled by the user. There will only be the movement functions on key press to start with and state-machines to check if it is alive, there could be more added on later. There will be values such as health and speed, this will be similar with enemy.

- **enemy (AI)** = this will be similar to player except will be controlled by state-machines to make a decision based on its circumstances. For example: *isTargetInSight? -> true = CHASE; (updated as per deltaTime)*.

3.3 Game Flow

The player can move with keyboard as well as accessing the pause screen (in-game) or main menu to start the game. There is so far two interactions that will happen between the health station or the enemy, both related to the stats of the player. The win condition is to survive for a certain amount of minutes against the enemy AI, while the lose condition is obvious, die before the timer is up by the enemy AI.

Because of this, The player need to be at a reasonable speed while not at the same speed as the AI (the AI will always be slightly faster since it relies on sight/hearing to spot, chase and attack the player), there will be walls as well to make the map a labyrinth-like-maze.

4.0 Mechanics

The player can do basic movement (WASD or UP,DOWN,LEFT,RIGHT) with one basic interaction (more can be added such as weapon/power ups) which involves the health station. The health station will provide health to the player if it is lower than its max health (100), this will trigger on collision with the player. The walls will be as it suggests, it will repel those who interact with it, this will be part of the entire map.

5.0 Graphics

The game is a 2D top-down, stealth, survival game that will incorporate 2D-art assets from the Kenny asset store.

6.0 Artificial Intelligence

The AI will be using a combination of state-machine and decision tree to perform 3 basic actions, depending on the conditions which is, WANDER, ATTACK while the in-game. These will be directly related on the player is seen or not, how far are they and the player's L.K.P (Last Known Position).

For Example:

- *isTargetInDistance? -> true = CHASE; (updated as per deltaTime to ensure it is true)*.

- *isTargetInDistance? -> false = CHASE up to L.K.P;*

The A\* pathfinding algorithm will be used to find the shortest path between two coordinates (the AI location and the location of the player when in its CHASE state.) This will be used for the enemy AI to find the quickest route to the player when it sees them, and if it loses sight of the player, it will remember the L.K.P of the player and find the quickest route. The algorithm will be using a grid to find the quickest route in this project.

Diagram of the AI states:

Diagram

Description automatically generated

7.0 Physics

Will be using Visual Studios and Unity 2D for this project, so will need to use variables and functions for the basic collision and interaction functions for the physics.

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 will only be one map that will be generated and cleared once the game has ended, then when rebooted, it will return to the same map before the start of the previous game (in other words, resetting the map).

9.2 Objectives

The player has to survive a certain amount of time against the enemy AI, when the player survives when time is out, they win, if not then the opposite.

10.0 Levels

N/A, there will only be a map as this game will not have levels.

11.0 Interface

11.1 Menu

it will be a simple menu which can be accessed while in-game to pause the game. These will be (for now):

- **Start** = when selected, the screen will clear then draw and update the game on a loop till either win/lose condition occurs.

- **Pause** = this will only be accessed during game, if the user presses a key or the pause button (either one), it will pause the game until the button/key is pressed again.

11.2 Camera

The camera will always be fixed in a top view position.

11.3 Controls

The controls for the player will be on keyboard to move, it can be either WASD or the arrow keys. There will also be some key shortcuts for the main menu and the pause screen.

14.0 Asset List

-Kenny for 2D art assets.

16.0 Technical Risks

There is minimal to zero risk with this project since it will be simplistic and easy to run and operate, however the only concern is making sure that there are minimal to zero bugs while ensuring no memory leaks and other errors.

17.0 Milestones

So far:

1. Set up A\* pathfinding algorithm
2. Set up state-machine class
3. Set up behaviour, mouse click controls, player classes
4. Set up enemy (AI) class
5. Set up Health station