### 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 | Fill out Development Environment |
| 1.2 | Fill out Game Overview |
| 1.3 | Update Game Flow |
| 1.4 | Fill out Mechanics |
| 1.5 | Add Bomberman example |
| 1.6 | Add State Machine flowchart |

2.0 Development Environment

2.1 Game Engine

**Unity Free Edition**

2.2 IDE

**Visual Studio 2019**

2.3 Source Control procedures

**GitHub**

2.4 Third Party Libraries

2.5 Other Software

**MSPaint**

3.0 Game Overview

* 1. Technical Goals
* **Working A\* Algorithm**
* **User Input**
* **AI Agents**

3.2 Game Objects and Logic

* **Maze**
* **Grid Object**
* **Moving Agent**
* **Change Spawnable Button**
* **Modify Maze Input Buttons**
* **Step Previous and Step Next Buttons**

3.3 Game Flow

**The game will immediately start with the player being able to move around by the user left clicking in the target position.**

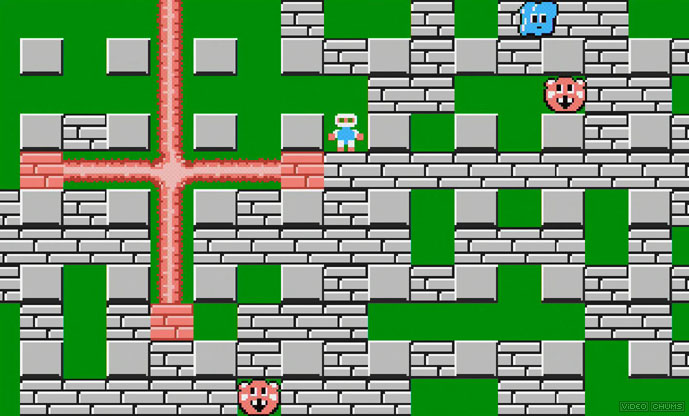
**The user will be able to spawn in whatever they want such as walls, enemy agents and maybe more.**

4.0 Mechanics

* **The player can be moved around by left clicking on a white tile.**
* **The player will avoid walls.**
* **The user can spawn in objects**
* **The user can modify objects.**

5.0 Graphics

**Basic top down 2D style. Characters will have simple animations. Graphics can be compared to bomberman.**



**Figure 1 – Bomberman example**

6.0 Artificial Intelligence

**Agents will use a state machine with 3 states (IDLE, PATROL and CHASE)**

* **On Idle nothing will really happen with the agent other than potential animation change.**
* **On Patrol a random coordinate around the agent will be selected and the AI will move to it.**
* **On chase the agent will travel to the target coordinates**

**Diagram

Description automatically generated**

**Figure 2 – Flowchart of Agent State Machine**

7.0 Physics

<if needed>

<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.>

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 |
| Default | 6 | 6 | 6 |  |
| Weapon | 5 | 7 | 7 |  |
| Weapon | 8 | 5 | 5 |  |
| Weapon | 5 | Possible 10 | NA | text |
| Weapon |  |  |  |  |

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?>

9.2 Objectives

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

10.0 Levels

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

11.0 Interface

11.1 Menu

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

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>

11.3 Controls

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

14.0 Asset List

<List all files needed, along with known attributes >

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>