**Design Proposal for Bomb It 112**

**Project Description**

**Name of term project: Bomb it 112**

**Description:**

Bomb it 112 will be a 16x16 grid-based game where players will place bombs that explode in a fixed cross. Bomb explosions destroy terrain and can kill other players if other players are caught up in the explosion radius. Destroyed terrain reveals powerups which can provide powerful abilities.

Players will start off with 10 lives and the goal is to deplete the lives of the opponents. Bomb it 112 will be single player before MVP where you will face off against 3 other AI players.

**Competitive Analysis**

This game is adapted from Bomb It 7, an online multiplayer game. The rules of the game are similar and how the game will play out will be similar.

I will be using different character models and sounds for my game to differentiate it. I will also add in other features such as different kinds of weapons and moving terrain to enrich the player experience.

**Bomb It 7 Image**



https://www.google.com/url?sa=i&url=https%3A%2F%2Fplayminigames.ru%2Fen%2Fgame%2Fbomb-it-7&psig=AOvVaw3Wkuvfhh\_epTHwmFhI5Kd9&ust=1636165242521000&source=images&cd=vfe&ved=0CAsQjRxqFwoTCMiN27iUgPQCFQAAAAAdAAAAABAr

**Structural Plan**

I plan to organise the project into different files that store different components of the project

1. Master.py (File containing all the code that runs the game) (Includes all the drawing of the board and the mode screens)
2. Character.py (File containing all the code that draws and models the characters)
3. Map.py (File containing code that generates the maze for the board)
4. Weapons.py (File containing all the code related to the weapons. Bombs first before others)
5. Sound (File containing all the sounds I will use in the game)
6. Images (File containing all the images I will use in the game)

**Algorithmic Plan**

The board used will be a 15 x 15 grid

Algorithmic complexity will come from 2 main components

1. Random maze generation
   1. I plan to randomly generate the destructible and non-destructible terrain in my game using maze generation algorithms such that players will have some room to move around even before destroying any destructible terrain
   2. I plan to use a few algorithms to generate my maze. I will google for the pseudo-code and write it myself
      1. Recursive Backtracking
      2. Kruskal
      3. Prim’s
   3. The generate maze will be a 2-d list of instances of my wall class where my wall class will contain attributes such as (row, col, destructible, color, etc.)
   4. This allows me to draw the walls at the maze wall coordinates easily
2. I plan to design an game AI that uses the minimax algorithm or DFS
   1. Most basic version:
      1. Optimal outcome: Drop bomb within 1 square of player
      2. Not optimal: Drop bomb > 5 square of player
      3. Neutral: Destroying terrain or picking up weapons
      4. I plan to make the AI do some random actions in between to ensure that the AI is not blindly rushing to the player to place a bomb
3. My game AI and maze generation will be supplemented by the following algorithms
   1. Depth first search
   2. Breath first search
   3. A star

**Timeline Plan**

* By TP0 (13 Nov):
  + Finish board drawing and basic player model
  + Finish bomb drawing and explosion radius animation
  + Finish part of wall class
  + Start on maze generation algorithm
  + Finish storyboard
* TP1 (18 Nov):
  + Start on AI algorithm
  + Finish maze generation
  + Import images and character models
  + Draw walls including destructible walls
  + Implement terrain reset (Cycle between different mazes)
  + Implement player model pictures (front, side, back view)
* TP2: (23 Nov):
  + Finish AI algorithm
  + Insert background images and sound
  + Have a working basic version of the game
* TP3: (01 Dec):
  + Add in more weapons and abilities
  + Add in revolving obstacles
  + Add in more sound for specific actions
  + Perhaps add in multiplayer?

**Version Control Plan**

I am using github to save all my code

Link: <https://github.com/Icyviolet23/Bomb_It_112_TP.git>

I will make a new branch or version tag every time I edit and merge to master once I have verified that the new branch worksA screenshot of a computer

Description automatically generated

**Module List**

* I am not using any external modules except for pygame for sound which was approved in the notes

**Storyboard**