Robo-Battle-Pigs

Game Design Document

# Game Overview

My game will involve 2-4 players competing head to head in turn based duels. Each player will control a Robot with unique ways of manoeuvring, manipulating and attacking across a randomly generated grid-based board. These robots will be given sets of instructions so that the players will have to try and predict each other up to 5 turns in advance.

# Gameplay Mechanics

Different robots will have different combinations of:

* Attack shapes (melee, leaps, beams, charges, Area of Effect, traps etc.)
* Movements shapes(diagonal, forward-backward, spinning moves, side-steps etc.)
* Utilities (disables, repairs, pushes, teleports, stealth, Damage over Time etc.)

As robots take damage they lose actions and have to take damage actions

The number of damage actions is representative of health and once 5 damage have been accumulated a robot is defeated and that player is out of the round.

Each turn players will have a limited time to determine their actions so as to help the game feel fast paced and exciting.

The game will also make use of an extensive set of modifiable rules to allow people to mess around with their friends.

Each round will take place on a random or preselected map with some tiles being obstructed and others being hazardous to create an interesting and varied gameplay experience.

This game will allow for both local and networked games.

There will be possibility for custom robots to be saved and used by players

# Software Design

|  |
| --- |
| Robot |
| name: what the robot is  health: int  actions: {‘F’:Action(), ‘H’:Attack(), …}  coords: [x, y]  facing: [0-3] always n % 4  is\_drawn: bool  is\_airborn: bool  sprite: Sprite() |
| act(key) returns tiles being affected and finishing tile and rotation  move([x,y])  dmg(int)  draw(screen, Board()) |

|  |
| --- |
| Action |
| Name: what the action is  shape: [[~x,~y, ‘effect’],[etc.]]  move: [~x, ~y]  turn: -1, 0, +1 +2  icon: pretty |
| do() returns relative tiles being affected and relative finishing tile and relative rotation of mech.  draw(screen) |
|  |
| Board |
| coords: [x][y] #where draw pos(x,y)  tiles: [x][y][type] pretty pictures  effects:[pictures] |
| where([x,y])  draw(screen) |

Map generation

Placing phase

Receives opponent turn

End screen

Determination phase

Waiting phase

Acting phase

Planning phase

Lobby

One Player is alive and all the others are defeated

Main menu

## Classes

The whole game takes place upon an isometric grid and so having a class to store the coordinates for drawing at and also the actual features of the map is useful. This class will be mainly used for knowing where to draw things but it will draw the board itself.

For the sake of abstraction the robot class will allow a type of robot to be defined by a constructor rather than individually hardcoding classes for all of my ideas. They will need to tell the game what a certain instruction does to them, and then they need to be able to change where they are and their health.

Another part of this abstraction is the action class, which allows for what the robots actually do to be abstracted and allow for customization. But they need to tell their robot what they do via relative coordinates and rotations and also how to display what they do to the screen.

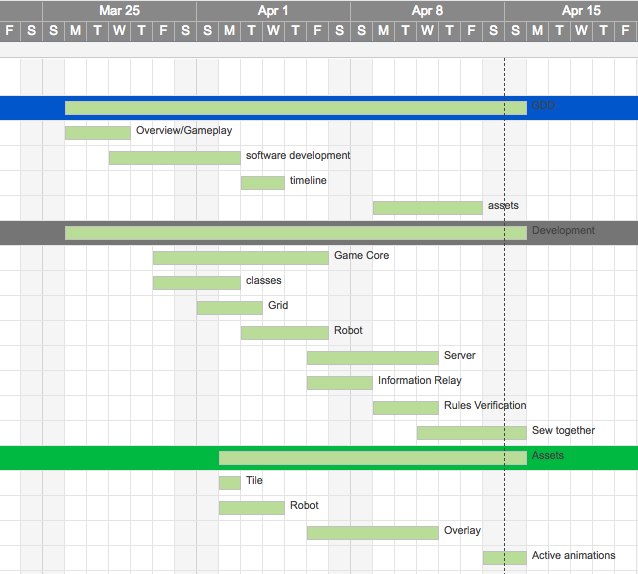
## States

Starting at the main menu a player will create a lobby for their friends to join, from there they can start a round. A round starts by a map being randomly generated and it is then distributed to all clients, they then choose where they start, from a specific section of the map. This is the placing phase. Then starts the main loop of the game.

Each turn starts with every player coming up with their plan for what to do in the next 5 steps, this will be timed to put some pressure on. If a player has only a partial plan, their plan will be finished with damage actions and any reciprocal actions and if they don’t have a plan at all they will repair. Once a player has completed their plan they enter the waiting phase until all other players have submitted their plans.

This leads to the determination phase, this is where all plans are evaluated by the server and the course of events is determined. This is then sent back out to all the clients and then they act out the 5 steps so that all the players can see what happened and then if one player has Won, the game ends otherwise it goes back to the planning phase. If the players want to play again they can return to lobby or they can go to the main menu.

# Timeline



# Assets

#### Graphics

* Tiles
* tile overlays(to show effects)
* robots(4 facings)
* active effects(damage, push, etc.)

#### Audio

* movement sounds
* damage roars
* music

I will create all graphics myself using GIMP

Almost all foreseeable sound will be from websites like freesound.com etc. these websites provide audio with more open copyright permissions so I will not be infringing copyright if I follow the producers wishes(sometimes detailed in description of an asset)