

Name: Devin Taylor

Game Title: Robots and Rubble

## Game Description

### Overview

The game takes place in a fixed, entirely visible, playing field. The player plays a ship attempting to salvage minerals in the territory of nasty robots intent on its destruction. The goal of the player each level is to collect a certain amount of minerals without being destroyed. The robots follow typically easy to predict AIs that are consistent to the type of robot. Most robots can be destroyed by being shot by the player, if this happens the robot leaves behind rubble, which will destroy any player or robot which runs into it. When a robot runs into rubble, both the robot and the rubble are destroyed. Robots are also destroyed when they collide with each other; this also leaves behind rubble. A player is destroyed when he runs into a robot, rubble, or a projectile shot by a robot.

In addition, robots can pick up the minerals that the player is attempting to collect, and get stronger in some way when they do (for the most part, they just move faster). Minerals continue to appear randomly throughout the map until the preset amount of minerals are harvested, or the player is destroyed. New enemies are also generated over time.

### Details

#### The Playing Field

A fixed, probably square, field. The player is destroyed if they collide with the edge of the playing field.

#### The Player

Moves like a ship in space. Commands: Thrust forward, rotate left or right, fire gun. Think of the ship in Asteroids. The gun is limited in range (about 1/6 of the playing field), and there can only be one player-produced projectile on the screen at once, encouraging accuracy.

#### The AIs

I have some basic AIs in mind, obviously some can be dropped if they prove too ambitious for the project.

Destroy – Heads towards the player's current location.

Wait and Destroy – If player is further away than X, wait in one spot, otherwise head towards the player's current location.

Block – Head towards the players **future** location (determine where the player would be if they continued along their current vector, and check where we would need to go to

intercept the player along that vector, and head in that direction). There could be a little randomness and simulated delay so the blockers don't react perfectly.

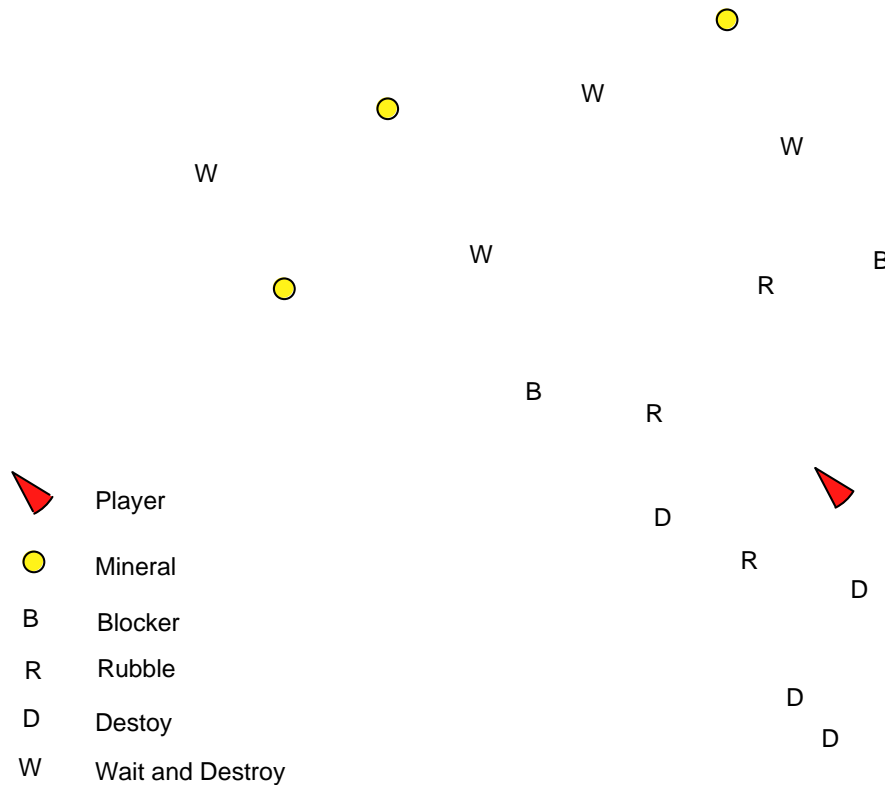
**Bouncer** – These move around the playing field in an initially randomly generated straight line. When they hit the edge of the playing field, they bounce off (like a ball in pong) and continue in a straight line. If they run into a mineral, the mineral is destroyed, and another bouncer is generated and moves in a line perpendicular to the current bouncer.

### Robot Abilities

**Shoot** – Some robots can shoot projectiles. They target the current location of the player. (Blockers might target where they think the player will be.)

**Sturdy** – These robots can't be destroyed by player fire, and must collide with another robot or with rubble.

**Sample playing field:** Player is being pursued by Destroy AIs and is about to be cut off by Blockers. Wait AI robots guard the remaining minerals. A skilled player can use the surrounding rubble as cover.



### Assessment

The main element of the game is rubble and robot collision. The limited usefulness of the player's weapon, and the "Sturdy" robots which can't be destroyed directly, should lead

to interesting tactics. The AIs should be simple enough to be able to manipulate into a collision, but diverse enough to provide the player with a challenge. Mineral collection gives the player a goal, forcing them to move around the map, possibly into danger.

### **Development Resources**

I see this game as a 2d game because I don't see any reason for a game to be in three dimensions is the players can't actually move in three dimensions. I have not researched free 2 dimensional game engines in detail. One possibility is the PTK engine (<http://www.phelios.com/ptk/>), which is a games framework (it gives us an API to work with for basic graphics, sound, and player input). It has a tutorial for getting set up on Visual Studio Express, which most of us will be using. It uses OpenGL and claims to be cross platform.

The sound requirements of the game shouldn't be too steep, and there are a number of ways to get good game sounds for non-commercial use. I'd prefer using 2-d sprite-based graphics, but there are not any reasons it could not be 3-d.