TITLE OF PROJECT:

AUTHORS:

PROJECT ABSTRACT

1. (a) Problem and Motivation:

The initial problem was how to implement a mechanic for the player to be able to change the weapon of the ship. I wanted the game to have many different unique weapons that were broken up into three categories of weapons, kinetic, laser, and missiles. I also wanted the player to be able to decide whether or not he/she wanted a specific weapon or not.

1. (b) Background:

In old arcade games, especially old shooters, it is common to implement stat changes through items that the player can pick up. These items may include health, shields, ammo, or other changes like weapons. In the old game Raiden Fighters, different colored orbs would sometimes spawn that the player could either pick up to change weapons or upgrade their current weapon. The orbs would change colors to allow the user to choose what kind of weapon they pick up or just go off the screen and disappear.

1. (c) Solution:

For our game I decided to use the same idea that Raiden Fighters used. I implemented it a little differently than they did though, because instead of having one weapon item that changes I made three kind of weapon items. Each weapon item is an item that contains a reference to an image, what kind of item it is, it’s collision box, and a velocity to which the item move every update. GameData is where all of the information is being stored and processed at. When certain events happen in the game, GameData will use a factory method to spawn in the items. The objects that are created can be predefined or randomly generated using a random number generator. This code snippet below is how each object is made.

**items.add((Item) weaponMaker.getWeapon("LASER", 400, 200));**

When the factory method creates the new item and it is rendered the object is stored in an array list within GameData. At this point the items float around freely until they are either picked up by a player or go off screen at which they are destroyed. In order for the player to pick up the items there is a check in GameData to check for collision between the item and the player ship. If there is a collision the information is saved in the ship object then the item is destroyed.

The ship object is able to process the information based on what the object is that is picked up and changes the current ship’s weapon to the appropriate weapon. Below is a code snippet of how this is done.

**//----collision: items vs friendly ships-----**

**for (int i = 0; i < this.items.size(); i++) {**

**//Rectangle[] hit = currentShip.getShipHitBox();**

**Rectangle hit = currentShip.getShipHitBox();**

**Item item = (Item) this.items.get(i);**

**if (hit.intersects(item.getRectangle1())) {**

**if (item.getItemType() == currentShip.getWeaponState()) {**

**currentShip.setLevelState(currentShip.getLevelState());**

**} else if (item.getItemType() >= 0 && item.getItemType() <= 2) {**

**currentShip.setWeaponState(item.getItemType());**

**currentShip.setLevelState(BASE\_LEVEL);**

**}**

**synchronized (items) {**

**this.items.remove(item);**

**}**

**}**

**}**

In the code above, if a ship intersects an item it first checks to see what the ship’s current weapon is. If it is the same as the weapon item picked up then the weapon level is incremented one from what the ship’s current weapon level is. If the item is different than the current ship’s weapon, the ship’s weapon type is changed to the item’s type and the weapon level is set to the base level.

1. (d) Result:

The end product is a weapon management system that operates similar to Raiden Fighters. The player is able to pick up item to either change what kind of weapon he/she has as well as upgrade the current weapon on the ship. The objects can also be lost if they go off screen.

1. (a) Problem and Motivation:

After I had the weapon types and levels mechanics working properly I needed a way to actually generate the bullets in the game. The system needed to be able to know what kind of weapons the ship had to generate the correct type of bullet. Another issue was that the system needed to be generic so it could be used by player controlled ships as well as machine controlled enemy ships. Yet another issue was that the bullets needed to be distinctly enemy or friendly so the player wouldn’t be damaged by his/her own shots.

2. (b) Background:

The idea behind having bullets is simple, shooter games need bullets or it isn’t a shooter. Without functioning weapons there wouldn’t be a game. I didn’t necessarily base my system off of any other games as much as the shooter genre itself.

2. (c) Solution:

My solution to bullet generation was to create a system that would take in data from a source and output a bullet object at a specific spot. I chose to make the bullet generation its own system to make is generic and reusable. I did this through the state design pattern which made the system loosely coupled with the type of ships firing bullets, reducing the workload for the team.

Basically, the system works by passing the

2. (d) result:

Do not simply write what you did or accomplished, but write HOW you achieved (technical contents) in terms of novel ideas, algorithms, design patterns, architectures, etc.

Q: **Abstract Contents**

I just wanted to clarify some what the Abstract should contain.

1. Is it fine to include some diagrams in the abstract?  
For example, can ER diagrams be included for when detailing the database design.  
  
2. Do sections of the abstract have to be labeled with each member's name?

For example, if Member A worked mainly on technical feature 1 does the section in the abstract about feature 1 have to be labeled with Member A's name.

A:

1. Yes, you may include diagrams, tables, pictures, etc.

2. No, you don't have to. I will refer to the final individual demo to assess one's contribution to the project.