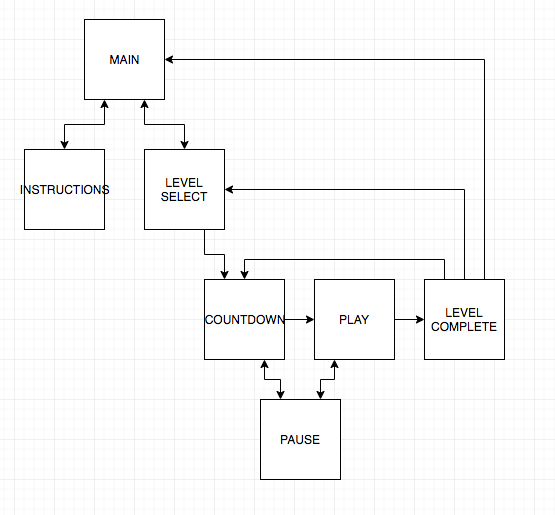
Ethan Frank Final Project Report

**Overview**:

I am creating an asteroid type game where you are a cow and fly around shooting enemies to beat levels. The project is an exercise in discovering proper game software organization practices and making an actually good proper game that unfortunately will never go onto the internet or somewhere cool. Also, the game has support for a PS4 controller.

**Flow Chart:**

High Level Overview:



Level Overview:

* Spawn enemies
* Player moves
* Enemies move
* Check collisions for all entities
* Check entities for removal
* Play death animations
* Remove if out of bounds
* If player dead, game over
* If level completed, move on

**Classes:**

**GraphicsMain** extends JFrame: This class creates the game window and adds the Graphics Panel to it.

**GraphicsPanel** extends JPanel: This class initializes Threads to play music and jiggle the mouse so the computer doesn’t go to sleep. It draws the background and calls the LevelManager’s draw function. It initializes the game’s levels and reads the human input from the input devices to send to the LevelManager.

**LevelManager**: This class handles the entire game, from buttons to the player to enemies.

**LevelData:** Holds a level’s name, spawn chances, criteria for traveling to the next level, and whether it has been unlocked or completed.

abstract **Character:** This class describes a game entity. Entities are the things that move around the screen during a level. This class comes with helpful methods such as *intersects()* and *outOfBounds()*. Characters have a position, velocity, angle, radius, health, an internal timer and more. All entities extend this class.

**Player** extendsCharacter: The player class is controlled by the player and can shoot. He has some health and when he dies the level is over. He loses health when hit by an enemy or an enemies bullets.

**Enemy** extends Character: Enemies come in many types and have different sizes, colors, health, behaviors, and point values. They lose health when hit by the players shots and when they have 0 health they implode. Most die when they go out of bounds. Some can shoot

**Shot** extends Character: Shots travel in a straight line and deal some amount of damage to other entities.

**Button**: Buttons are orange and rectangular. The update method causes them to highlight when the mouse is over and calls their ActionListener when they are clicked.

**GameController**: Handles human input. Depending on if a controller is plugged in or not, this class will set it’s public instance variables to either data from the controller or the mouse/keyboard so that GraphicsPanel can pass it to LevelManager.

**ClockListener**: Has a method that gets called by a timer in the GraphicsPanel

**Enums:**

**SoundEffects:** Holds all the sound effects that can be played. Has a method to play them.

**State:** Describes a state of the game. The game is always in a state. Some states have buttons, which are all updated and drawn each tick of the game.