Design Task

Artistic Design

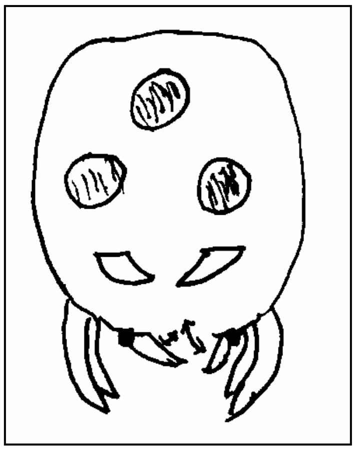
Metroid Pinball

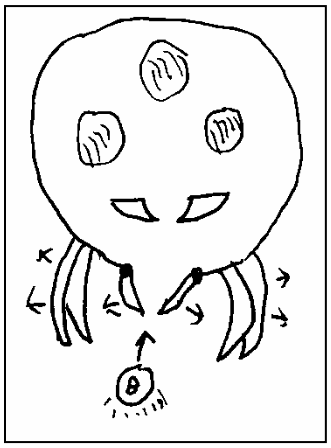
The main theme for this game will be Metroid, a popular video game series created by Nintendo. The hero the game features a "ball-mode" so the pinball game will be designed to resemble certain aspects of that. The main playing field will be inside a Metroid and its teeth will serve as flippers. The game will also use background music and scores from the original games and some sound effects of the games as appropriate (and available). For example when the flipper hits the ball, an explosion sound will be used or when the ball is launched into the field the "Rocket launched" sound will be played. The game will start by shooting the ball into the playing field through the mouth of the Metroid between the two teeth/flippers so there is no special "launch way" for the ball. The "Start Screen" and the "End Screen" will also be modeled after the corresponding screens in the game.

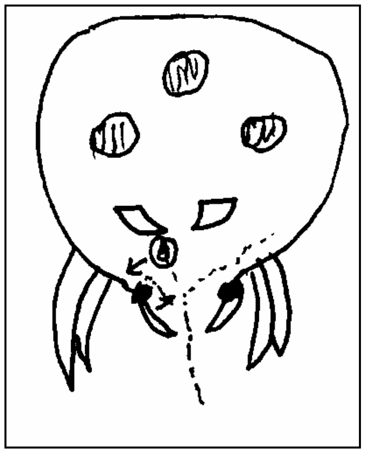
Storyboards

Ball is launched into the field of play

The only opening in the game field is the Mouth of the Metroid. Before a new ball is launched the Metroid will automatically open its mouth. After that a new ball will be launched into the game field through the opening.

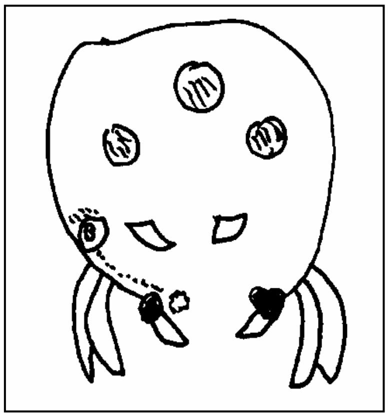


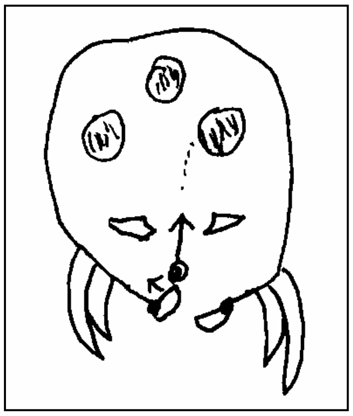


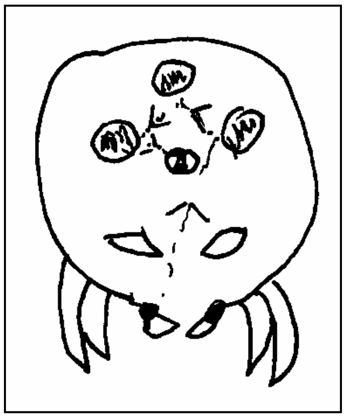


Flipper hits the ball

The inner teeth of the Metroid serve as flippers. There are also three “energy orbs” that serve as bumpers that will accelerate the ball upon collision. They start out in green color tones but will turn red if they are hit by the ball repeatedly.

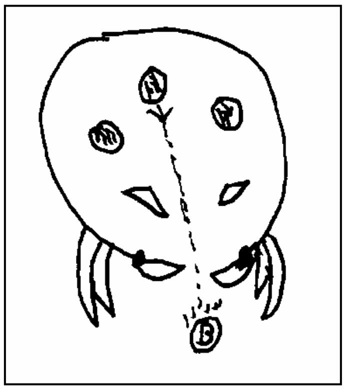


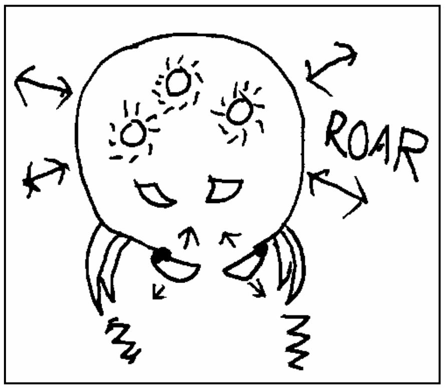


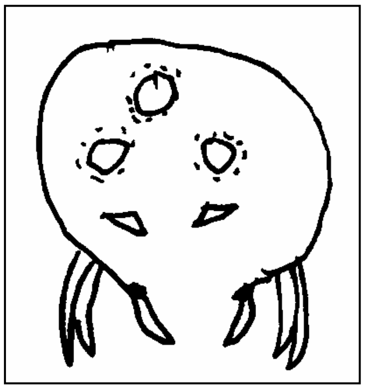


Ball leaves the field of play

When the ball leaves the game field, the whole Metroid will turn angry. It will wobble and shake its tentacles accompanied by some appropriate sound effects. The three “energy orbs” will glow red as well. Afterwards the Metroid will be sad and the mouth will open up - ready for a new ball.

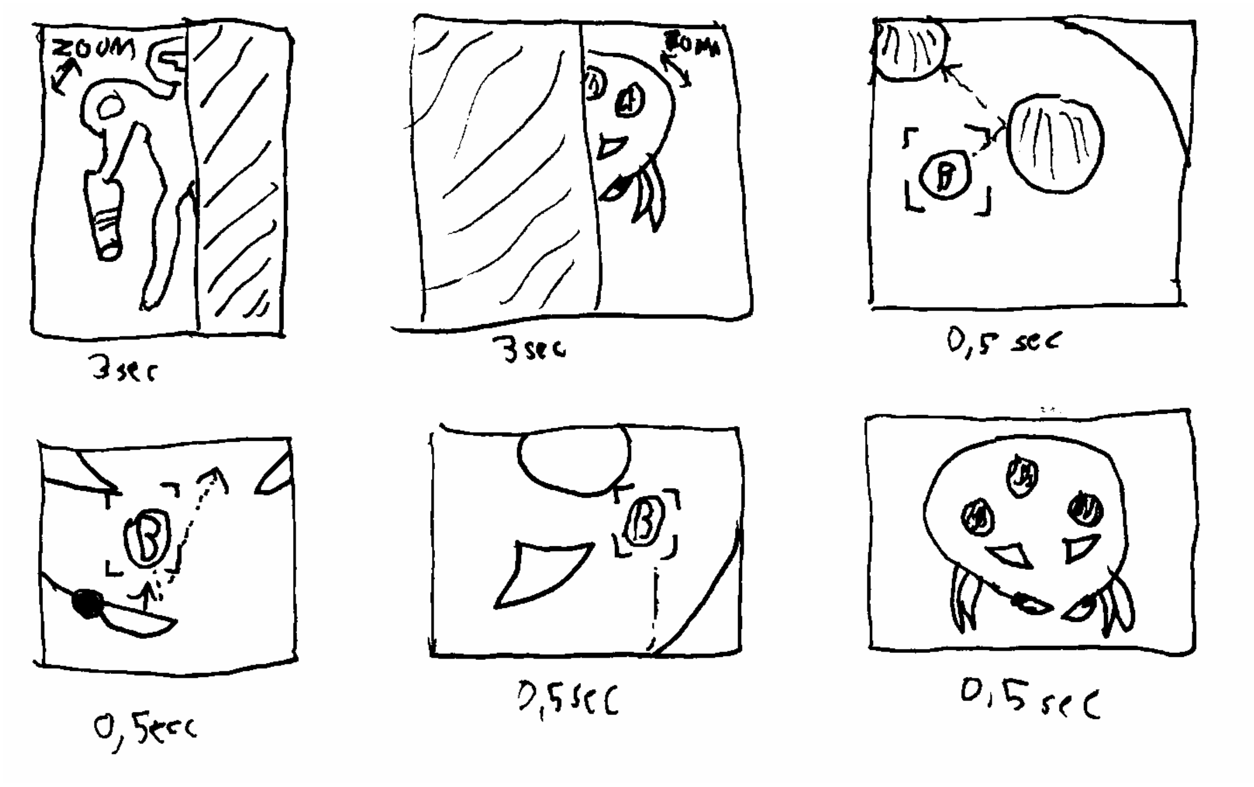






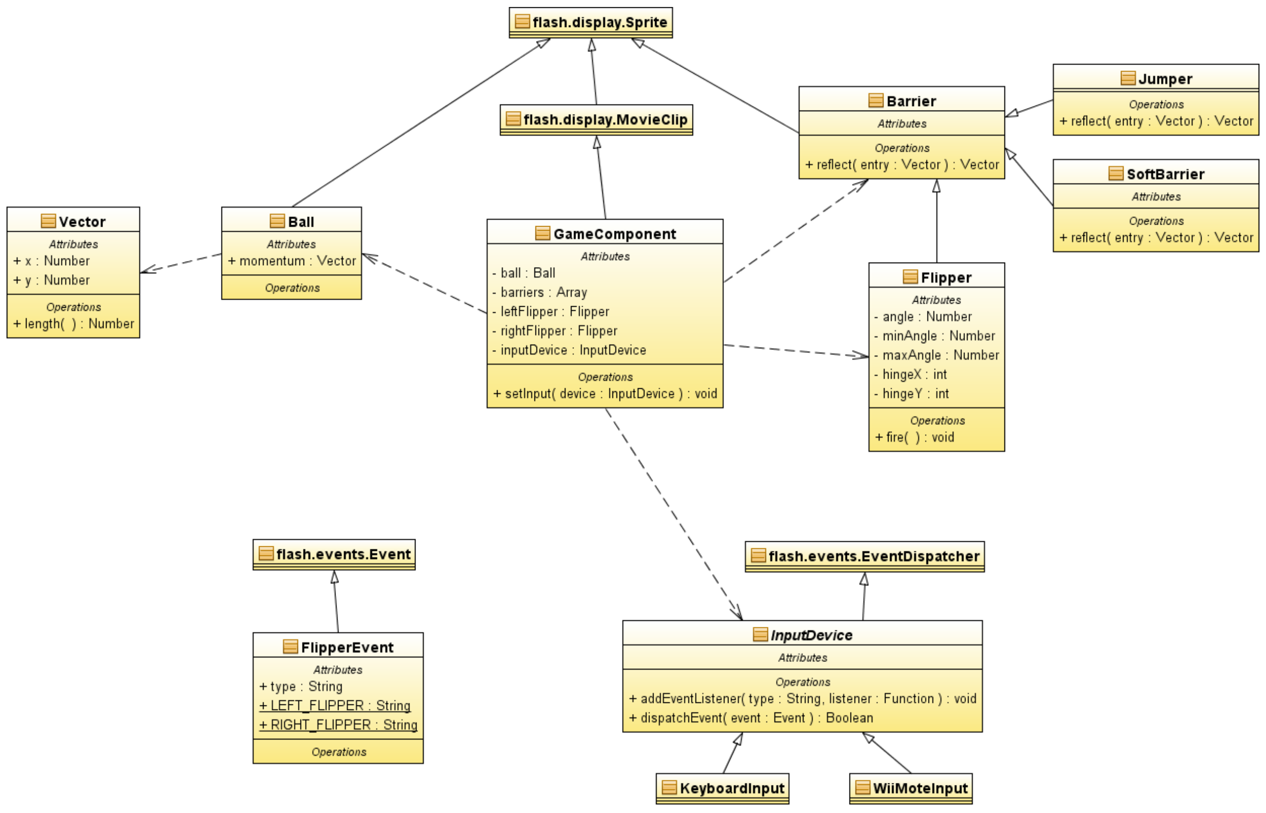
Introductory Animation

The introductory animation will start with slowly zooming in Samus Aran, the heroine of the Metroid series. The right-hand side of the stage is blacked out at this point. Afterwards the camera switches to the Metroid. This time the left-hand side of the stage is blacked out. This comes down to the “hero” being on the left-hand side while the “enemy” is on the right-hand side. The scene uses the “Shot reverse shot” technique to indicate a confrontation and the slow zoom is intended to create tension. Afterwards short clips of the game itself follow in rapid succession. These clips are however zoomed-in on the ball so the player cannot see the whole field of play. Only the last image in this sequence of clips will show the player a short glimpse of the whole field. This scene uses “Fast cutting” for the overall sequence to imply a fast and exciting game. Each individual clip is a “Tracking shot”.



Technical Design

Architecture



This class diagram depicts the main classes that will be used while playing the game. GameComponent connects most of these classes. It will display the playing field, barriers, the ball and the two flippers.

Barriers are the components that the ball can collide with and bounce off. In absence of a full-fledged physics engine, each Barrier knows how to reflect the ball in case of a collision. This allows for simple playing field borders that only change the direction of the ball but also “jumper” or “soft material” barriers that will accelerate or decelerate the ball upon collision.

The GameComponent itself will not directly respond to user input put only flipper events. InputDevice is an abstract class responsible for listening to user input and translating it to flipper events. Specific implementations may listen to keystrokes, mouse clicks, voice recognition, network input, wiimote gestures, etc.

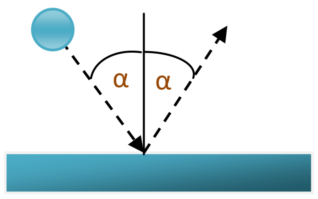
Simple Ball Physics

The current momentum of the ball is defined as a 2D vector. The direction of the vector corresponds to the direction of the ball and the length of the vector corresponds to the speed of the ball.
Gravity is a constant force that continuously accelerates the ball toward the bottom of the game field.

This can easily be simulated by adding a gravity vector to the momentum vector of the ball after each frame. The amount of gravity can then be fine-tuned by changing the length of the gravity vector until the desired behavior is achieved.
Spin of the ball is not considered here because for this application the spin and how it changes the reflection angle upon collision is negligible and would unnecessarily complicate things.

Simple Reflection

The math behind the deflection of the ball is simple if you know the normal of the surface on the point of collision. The angle between the normal and the entry vector equals the angle between the normal and the escape vector. The length (ball speed) of entry vector and the escape vector is equal as well.



The problem in calculation the escape vector however is calculating the normal at the point of collision. If the surface is even, the normal is known immediately, but if the surface is uneven, the normal has to be estimated by using the point of impact and some adjacent points to determine the angle of the surface at point of impact.

Tangible Interaction using a WiiMote

The controls of a pinball game are usually limited to two buttons, one for each flipper. Those two flippers could be controlled by a WiiMote with Nunchuck that is connected to a computer via Bluetooth. It would allow a more fine-grained control on how fast the flipper moves and thereby how hard the ball is hit on impact. This can easily be integrated by adding an additional property to the FlipperEvent and a little bit of additional logic in the GameComponent. Flash does not support WiiMotes out of the box but there are open source projects like wiiflash that will allow you to access WiiMote button and rotation states from within ActionScript. Gesture recognition still has to be implemented however which may pose a problem.