Gesture Based UI Development Project

Bowling Bonanza

Team Members: Jeremy Yon, Daire Ni Chathain

Purpose of Application

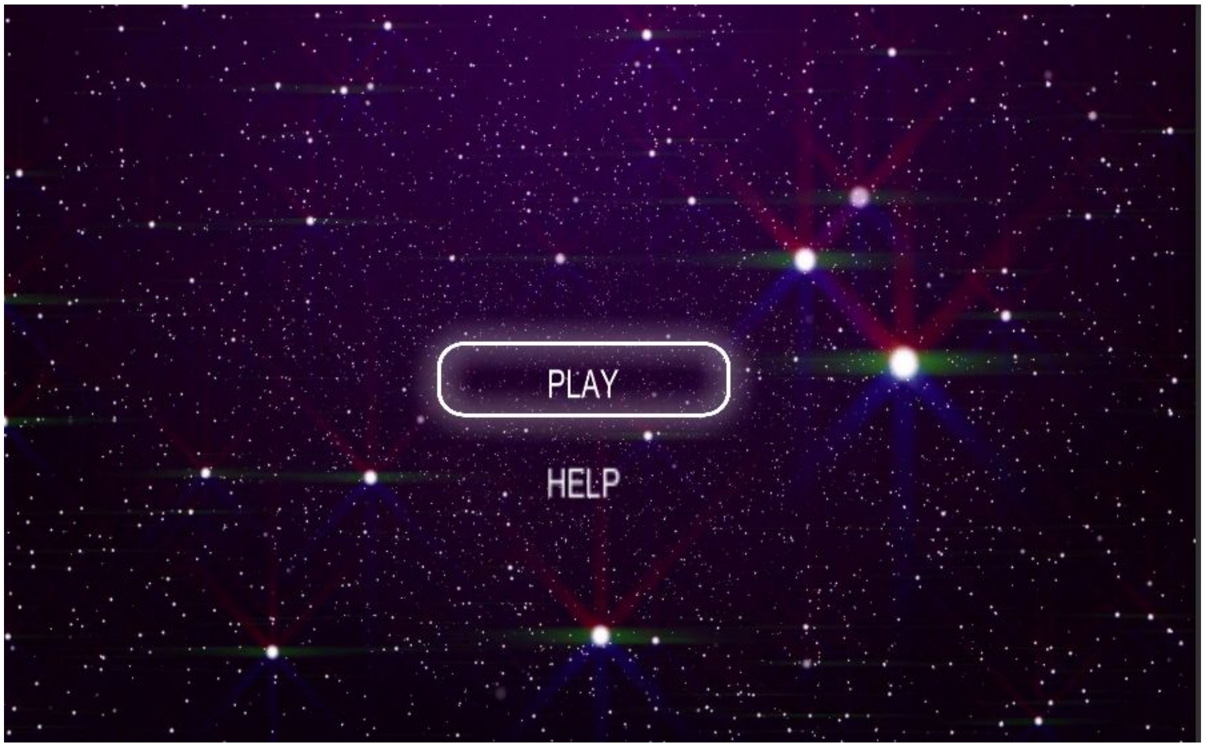
The purpose of this application is to explore the recreation of a classic game using innovative gestures and UI development. Aiming to drive our innovation through the use of intuitive gestures, we consider this application an experimentation process of sorts. We investigated how various hardware components, combined with intuitive gestures can help breathe new life into a retro game. The game we chose to redesign was a traditional Bowling game. Examining the natural movements and gameplay associated with the sport we sought to capture its essence to create a novel gesture-based game.

Application interface

Menu system

Start scene

- To start the game the user is first met with the Start scene. From here the user may click one of two options.



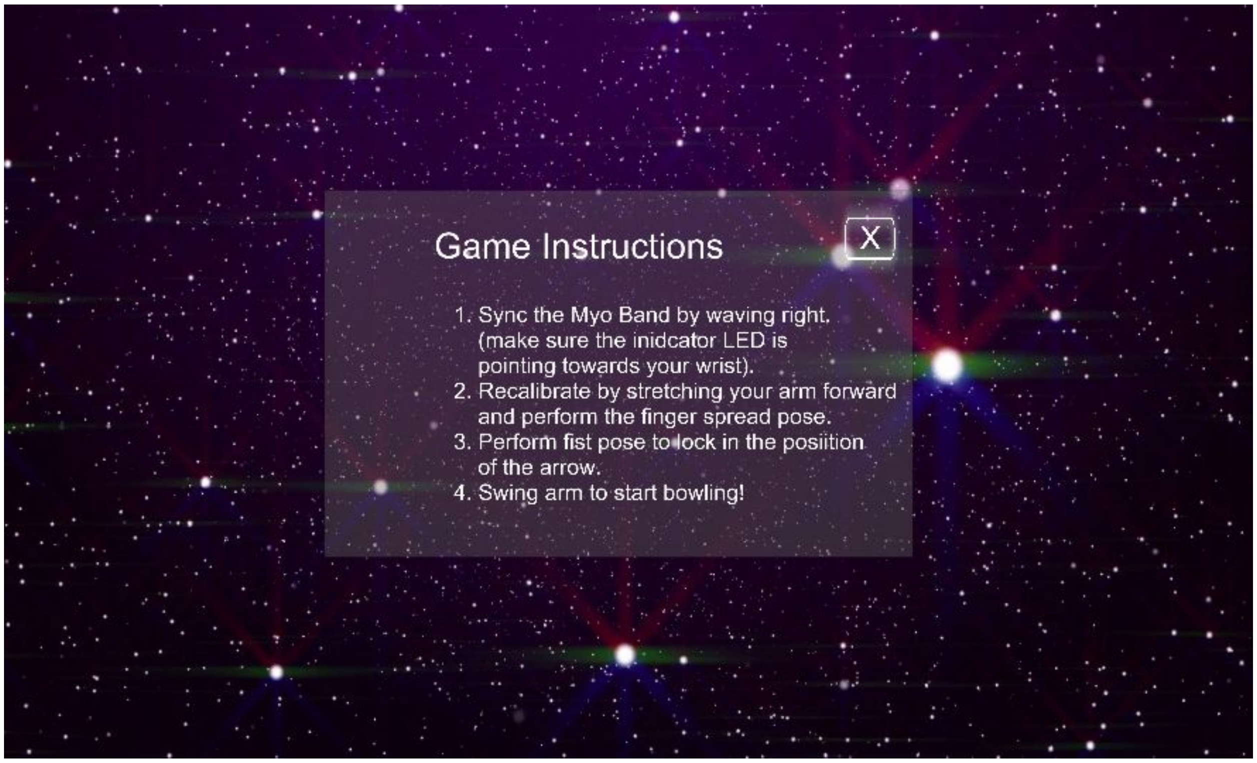
1. **Play**
   a. The play button begins the gameplay.
2. **Help**
   a. The button leads the user to a tutorial screen.

Tutorial scene

- Once the user clicks the help button they are brought to the tutorial section of the game. Here a description of the game play is presented to the user.

- The Instructions include:

1. How to use the MYO hardware
2. How to calibrate hardware using gestures
3. How to perform the bowling gestures



Game Instructions

1. Sync the Myo Band by waving right. (make sure the inidcator LED is pointing towards your wrist).
2. Recalibrate by stretching your arm forward and perform the finger spread pose.
3. Perform fist pose to lock in the posiition of the arrow.
4. Swing arm to start bowling!

Research Phase

During the research phase, we investigated real life bowling hand motions that affect the path and the physics of the bowling ball. We studied the intricacies in the gestures of the sport, from its swinging motions which have a large range of motion, to the subtle wrist rotations that have huge implications on the real life physics of the ball.

Bowling Physics

The optimal trajectory of a bowling ball is a curved path where it strikes the pins at an angle. Striking the pins at an angle improves the chances that there will be a "strike" in which all the pins are knocked down. Analyzing and understanding the physics and underlying factors that influence how a bowling ball curves was key to creating meaningful gestures.

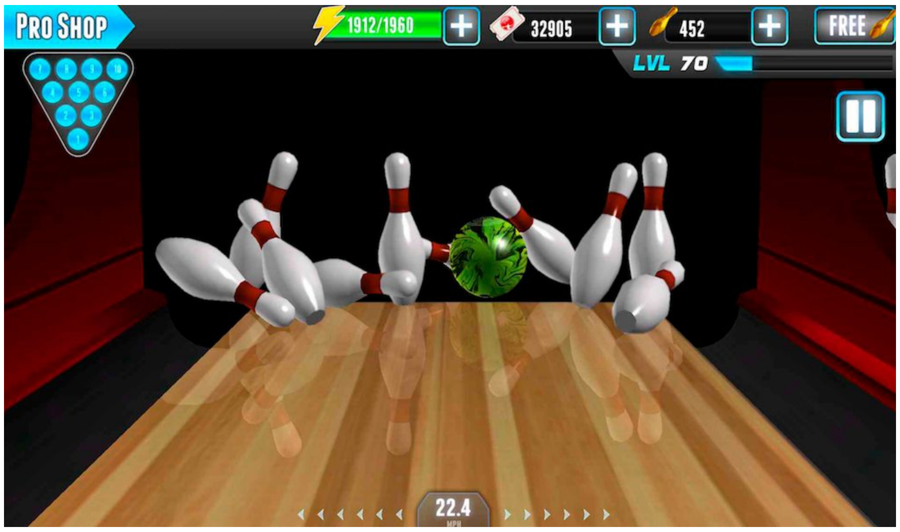
Giving the user the ability and degree of control needed to take a realistic shot was a focal point during the application's development. If the ball follows a curved path, it will be able to strike the pins at a greater angle than a bowling ball that travels in a straight line. Therefore, controlling the degree of curvature was essential to making the gaming authentic to a real world bowling experience. [1]

- [1] https://www.real-world-physics-problems.com/physics-of-bowling.html

Inspiration + Resources

We researched bowling games that were available and the controls that were being used to generate accurate physics data to mimic friction, gravity and push forces on a bowling ball. These games included mobile bowling games on smartphones that used swipe gestures to create spin and forward push forces on the bowling ball. We analysed how different swipes affected the path of the ball, both in real life and in games.

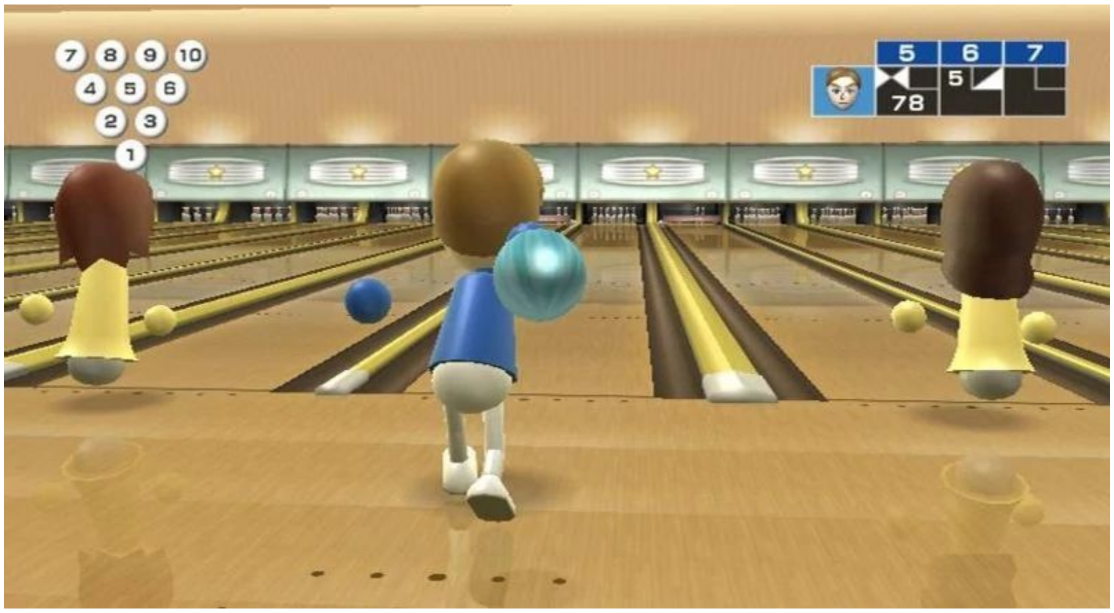
Bowling 3D



Bowling 3d is a popular mobile application available on both iOS and Android devices. The key elements we identified that contribute to the game's success are the following:

- Intuitive interface
    - Simple to navigate straightforward
- Gestures
    - The primary gesture evident in this application is the swiping action to control the ball roll forward.

Wii Sports Bowling

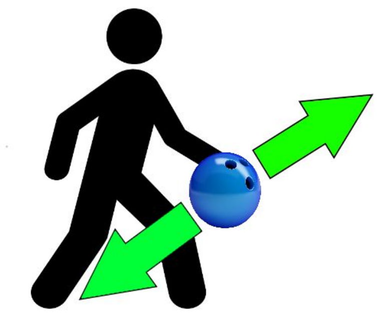


Wii Sports Bowling was developed and published by Nintendo for the console. WE noted the following key elements during our game research and investigation phase:

Game Control + Gestures

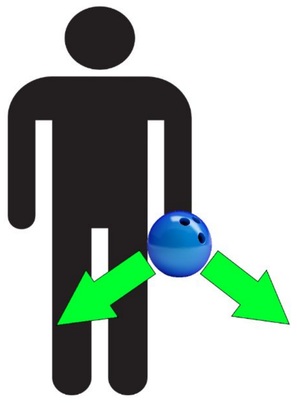
- The primary gesture evident in this application is the swinging action to control the ball roll forward. This is achieved through the use of the wii controller and nunchuck.

Arm Swing Forward



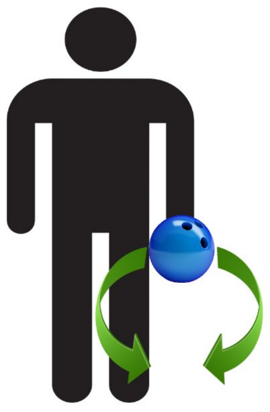
In a game of bowling, the forward swinging arm generates the force required to push the ball from the start of the bowling alley to the pins at the end. The player starts by holding his arm up in front of him perpendicular to him, swings the ball back, and then rolls the ball by swinging the ball forward and letting go of the grip on the ball. This is a core spatial gesture in a game of bowling, and has to be represented in the most natural way possible.

Arm Swing Direction



The arm swing direction is a spatial gesture that will control the straight line path of the ball in a game of bowling. Swinging the arm left will direct the ball towards the left side of the alley, while swinging the arm right will direct the ball towards the right side of the alley.

Wrist Rotation



The rotation of the wrist is the third spatial gesture that manipulates the forces acting on the ball. The rotation of the wrist generates a spin on the ball, which in turn creates an arc in the trajectory. Spinning the ball is necessary to get the optimal trajectory that will lead to a strike.

Gripping the ball

The fist gestural pose is used to mimic the gripping of the bowling ball. The ball is gripped before the start of the swinging motion, to inform the program to recording the spatial data.

Next, we decided that the gestures that we were to use in this Gestures and UI project would have to copy real life arm and hand motions. This will create an immersive experience for the user, as the gestures used will feel natural and as similar to real life as possible. Next, the physics on the game can be recreated so that the ball path will correctly adhere to the natural law of physics. We identified all the gestures that will be required in the development of the game, and divided them into 2 different types of gestures: spatial gestures and gestures using poses.

Interaction Design

Design Goals

We strived to create a responsive, interactive gesture based application. To do so, we kept the following in mind throughout the interaction design and development:

1. Create a user experiences that is context-aware
2. Keep interactions simple, easy to learn master
3. GIve constant feedback so user know whats happening
4. Ensure input requires the least overall effort & input should not be changed

Innate vs Learned Gestures

During the research phase, we identified the various types of gestures essential to the creation of a successful gesture based game. This application contains a combination of innate and learned gestures.

Innate gestures relate to gestures the user intuitively knows based on their understanding of the world. An example of innate gestures is evident in the inclusion of the "forward swing gesture". This action is natural, the user instinctively performs this movement when trying to emulate a bowling roll.

In contrast to innate gestures, learned gestures are physical actions that must be taught before use. They are specific and often unique to an application's use case. In the case of this game, the "sync gesture " is considered a learned gesture. The principles of interaction design state that if a gesture is not intuitive a detailed explanation should be provided. For this reason, we included a descriptive instruction of how to conduct the pose in the game tutorial.

Static Gestures

Static gestures can be defined as fixed position gestures the user must match to be recognized as meaningful. Static gestures are evident in this app in the use of the Sync Gesture. The user must match the gesture exactly for the application to identify and process it.

Dynamic Gestures

Dynamic gestures are defined movement that allows the user to directly manipulate an object. IN contrast to static gestures, dynamic gestures capture a range of movement. A example of dynamic gestures are incorporated in this application is the "grip ball" gesture. This dynamic gesture allows the user to directly control an action in the game.

Continuous Gestures

Continuous gestures encapsulate the prolonged tracking of movement where no specific pose is recognised. This general movement is used to interact with the application. Use of continuous gestures in this application is evident in the 'arm swing forward' gesture. This dynamic gesture tracks the movement of the arm for an extended period of time to capture the key bowling action.

Hardware

During the research phase, we investigated various hardware that would meet our requirements in performing the gestures mentioned above. We wanted something that was light, easy to use, responsive, able to provide accurate data, as well as a library that could be integrated seamlessly into our project. We carefully considered the hardware that was available to us, namely: Myo bands, Leap motion controllers, Microsoft Kinect.

Leap Motion Controller

We explored the option of using the Leap Motion Controller during the development of this project. This piece of hardware consists of a small USB device that can be placed on the desktop or mounted on the forehead. It supports gesture capture for forearms, hands and fingers.

Microsoft Kinect

Kinect is Microsoft's motion sensor add-on for the Xbox 360 gaming console. Kinect features a natural user interface that allows users to interact intuitively with a game system by distinguishing gestures, voice commands, facial characteristics, skeletal data and full body motions. Gaming applications are able to recognize individual players with the help of skeletal data and identify each player by name and other features.

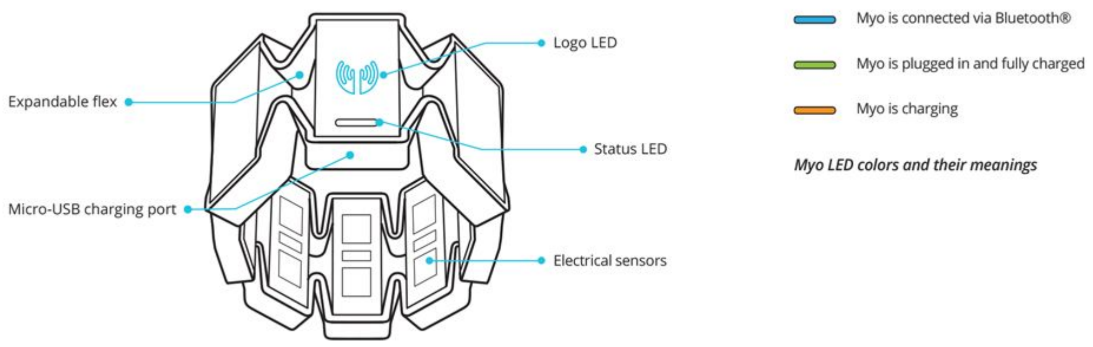
The Kinect contains three vital pieces that work together to detect your motion and create your physical image on the screen: an RGB color VGA video camera, a depth sensor, and a multi-array microphone. The camera detects the red, green, and blue color components as well as body-type and facial features. It has a pixel resolution of 640x480 and a frame rate of 30 fps. This helps in facial recognition and body recognition.

The depth sensor contains a monochrome CMOS sensor and infrared projector that help create the 3D imagery throughout the room. It also measures the distance of each point of the player's body by transmitting invisible near-infrared light and measuring its "time of flight" after it reflects off the objects. The microphone is actually an array of four microphones that can isolate the voices of the player from other background noises allowing players to use their voices as an added control feature.

These components come together to detect and track 48 different points on each player's body and repeats 30 times every second.
[https://www.jameco.com/jameco/workshop/howitworks/xboxkinect.html]

MYO Band

The Myo Band is a Gesture Control Armband, that allows the user to wirelessly control technology with hand gestures. It measures electrical activity from your muscles to detect gestures made by your hands, and also senses the motion, orientation and rotation of your forearm using EMG sensors. It works with Mac, Windows, iOS, and Android devices through Bluetooth Smart.



Myo LED colors and their meanings

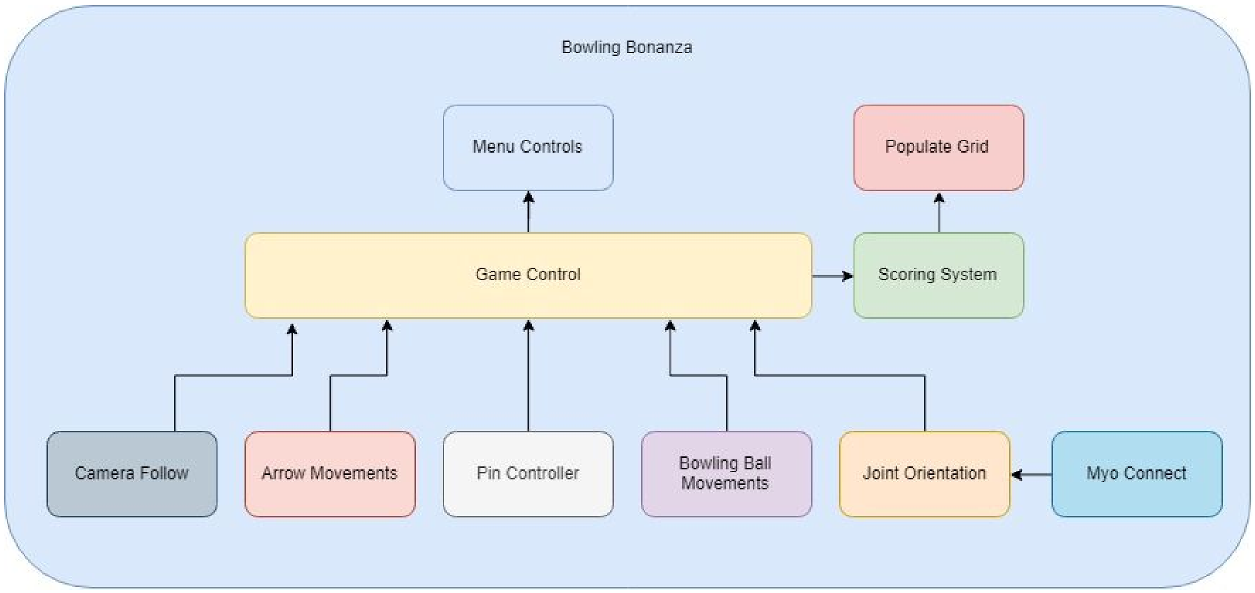
Hardware Selection Process

We decided that the Myo Band would be the ideal tool for the development of this project. Compared to the Leap Motion, the MYO armband allows users to control a computer at distances greater than a few feet. Furthermore, because it is detecting motion within the arm, control can be wielded during a variety of tasks whether facing the computer or not. The proximity requirement of a Leap controller led us to rule it out as a viable hardware option as the bowling gestures needed in this application require a wide range of motion.

While the Kinect allows for a sufficient range of tracking, the MYO band’s EMG sensors yield a greater accuracy for arm movement recognition.

Following an analysis of our hardware options, the MYO band exhibited the most attractive features in terms of capturing the gestures needed in this application.

Architecture Overview



Libraries used:

- Myo-unity plugin
- Myo Connect

Game Engine

- Unity 2018.3.5f1

Bowling Ball Movements

This is the main class that controls all movements of the bowling ball. It is responsible for all the physics forces acted on the ball, such as the spin of the ball, the throwing direction of the ball, and the speed of the ball. It handles the resetting of the ball and pins back to their original positions after each throw.

Camera Follow

This class controls the movements of the camera. It makes sure the camera tracks the position of the ball and follows it as the ball is bowled down the alley, and stops the tracking when the bowling ball hits the pins.

Arrow Movements

This class controls the arrow that will determine the starting position of the ball. The arrow will move from left to right off the bowling alley, and the user will have to “lock in” the position of the arrow before every throw. The movement will resume after every throw.

Joint Orientation

This class is extracted from https://github.com/thalmiclabs/myo-unity sample file, which contains the code that handles all arm movements of the user. It communicates with Myo connect to retrieve Myo Band data. It is able to process the spatial and gestural data of the Myo Band from Myo connect, and represent the data in the movements of the “arm” that controls the swing and direction of the ball. It captures the fist pose to lock in the direction of the arrow, and then converts the position and rotation of the arm and wrist into force values of the movement and spin of the ball.

Myo Connect

Myo Connect is an application used to sync and hold the Bluetooth connection between the computer and the Myo Band. It is needed to pair the band to the computer, and then communicates with the band using Bluetooth Smart. It obtains the physical postions and EMG data of the Myo Band and converts it into spatial and gestural data, to be used by the Joint Orientation class.

Pin Controller

This class handles the interaction between the pins and the other objects of the game. It detects the collision between the ball and the pins, and causes them to fall. It detects and sums up the pins that have been knocked down, and then sends this information to the game controller.

Game Control

This is the core class of the whole program. It handles the flow of the game, from start to finish. It ties up the classes used in the game, starting setting the pins and ball in its original position, to the whole process of detecting the data from the Myo Band, to initiating the ball movements, and finally getting the current score and sending it to the scoring system.

Scoring System

This class contains the formula to calculate the scores in a game of bowling. It sums up the number of pins that have been knocked down in a frame, and make necessary adjustments if its a spare or a strike. It keeps track of the score from all the frames and sends the data to the Populate Grid class so that it can be displayed on the canvas of the program.

Populate Grid

This class populates a grid on the canvas of the game with the current score, and updates it after every throw.

Test Plan

The Test Plan has been created to communicate the approach to testing in the creation of this gesture based application. It includes the objectives, scope, test cases. The scope of the tests range from user interface elements such as menus/buttons to testing game functionality and mechanics to user experience.

Tactics + Strategy

The tactics employed in this test phase consist primarily of system integration testing and user acceptance testing.

Methodology:
System and Integration Testing

Participants:
THe developers of this project conducted the systems and Integration testing.

System Testing:
For this phase of testing we chose to test the key functionalities of the game. Each test case has an action and desired outcome.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual result	Pass
TU01	Test Start Menu	1. Start game 2. Wait for start menu to appear	n/a	Start Menu menu should appear first when game starts	As Expected	True
TU02	Test Play Button	1. Start game 2. Click play button 3. Successfully executes on click function	n/a	Button successfully executes onClick method	As Expected	True
TU03	Test Help Button	1. Start game 2. Click help button 3. Loads tutorial scene	n/a	Button successfully loads tutorial scene	As Expected	True
TU04	Test MYO sync gesture	1. Ensure myo is connected 2. Start game 3. Click start button 4. Perform sync gesture	Input data from MYO	MYO successfully syncs and vibrates	As Expected	True

TU05	Test MYO grip gesture	1. Ensure myo is connected 2. Start game 3. Click start button 4. Perform grip gesture	Input data from MYO	MYO successfully recognises grip gesture and bowling ball locks position	As Expected	True
TU06	Test MYO swing forward gesture	1. Ensure myo is connected 2. Start game 3. Click start button 4. Perform swing forward gesture	Input data from MYO	MYO successfully recognises swing gesture and rolls bowling ball	As Expected	True
TU07	Test MYO swing direction	1. Ensure myo is connected 2. Start game 3. Click start button 4. Perform swing forward gesture with left direction	Input data from MYO	MYO successfully recognises swing gesture and rolls bowling ball in left direction	As Expected	True
TU08	Test UI score update	1. Ensure myo is connected 2. Start game 3. Click start button 4. Perform swing forward gesture 5. UI score updates	Input data from MYO	Score is update to reflect number of pins hit	As Expected	True
TU09	Test bowling ball and pin collision	1. Ensure myo is connected 2. Start game 3. Click start button 4. Perform swing forward gesture 5. Hit pin with bowling ball	Input data from MYO	Collision between bowling ball and b pins	As Expected	True

User Acceptance Testing/ Beta Testing

The purpose of acceptance test is to confirm that the system is ready for operational use. During these acceptance tests, end-users (customers) of the system compare the system to its initial requirements.

Participants:
Beta & Alpha Testers: Consisting of fellow college students and friends/family.

Methodology:
The user acceptance testing involved getting the participants to test the game. They then completed a feedback survey which outlined their thoughts on the game and suggestions for

improvements and any bug/defect encountered. In keeping with the principles of interaction design, we aimed to gather as much feedback relating to gestures and whether the user found the interactions simple, intuitive and easy to learn.

Results :

The feedback received from the user acceptance phase included:

- User Experience suggestions
- Change gesture from open hand to first to emulate gripping ball action
- Include Help Section/Game Tutorial
- Several bugs were identified . e.g faulty button/gesture not recognised

Following analysis of the feedback received from testers, we remedied the bugs identified in the game. This included an error where the bowling ball would at times not fire after the swing forward gesture. We also altered the ‘grip bowling ball’ gesture from an open hand to a fist following users suggesting the later gesture was not natural and somewhat confusing.

Conclusion

What we have learnt

1. Gestures used to create an immersive virtual experience
   - After working on this project, we learnt that Gestures can be used to simulate real life actions. We learnt that using tools such as the Myo Band can help us mimic a game of bowling as close as possible, for example, the ability to swing your arm to roll the ball.

2. Effectiveness of Myo Band
   - The Myo Band is a great tool to track hand gestures. The myo-unity plugin used in our project allowed us to sync the band to the computer with ease and to hold its connection without any interruptions. The process of retrieving the spatial and data from the Myo Band was relatively easy and accurate.

3. Test Plan essential to iron out bugs
   - After the implementation of each new class, we tested the actual results against the expected results, to see if the test case would pass. This helped us identify bugs and fixed them before we moved on to the next implementation.

Recommendation

1. Incorporate more gestures
   - More gestures could be added, for example releasing the ball when the gestural pose is changed from fist to a resting pose, to make the game more intuitive. Accelerometer data could be use to change the speed of the ball depending on the speed of the swinging arm.

2. Multiplayer mode
   - Multiplayer mode could be added to the game, so that multiple players can play one after the other after each frame, just like in a real life game of bowling. Multiple arm bands could be set up so they work on one computer.

3. Incorporate different technologies
   - In this project, we used to Myo Band to handle the gestures needed to complete it. For future development, we could possibly explore different technologies, such as the Kinect to track the movement of the whole body. Voice commands could be used to navigate through the options and main menu.