CMPT 362 Fall 2025 Group 30

Members:

Ario Katchooi

- Website creation & hosting
- Sensor research

Branden Nero

- Presentation Creation
- Open GL ES research & implementation

Cameron Lee

- UI drafting & implementation
- Firebase research & implementation

Jonathan Osuji

- UI draft refinement
- UI implementation

Section One

Pitch Refinement

Section One: Pitch Refinement

- App Theming & Name
- Online Leaderboard

Section Two: App Demonstration & Implementation

- Accessibility Features: Vibration and Sound
- Manual Controls using Motion Events
- Inertial Navigation using Sensors
- Rendering using Open GL ES
- Orientation Field of View Change
- Leaderboard using Firebase
- User Interface Views
- Model View Viewmodel for App Data

Section Three: Show and Tell 2

- Goals

App Theming & Name

- Our app will be aquatic themed
- The app's name is Aquatic Exploration
- The player character will be a submarine

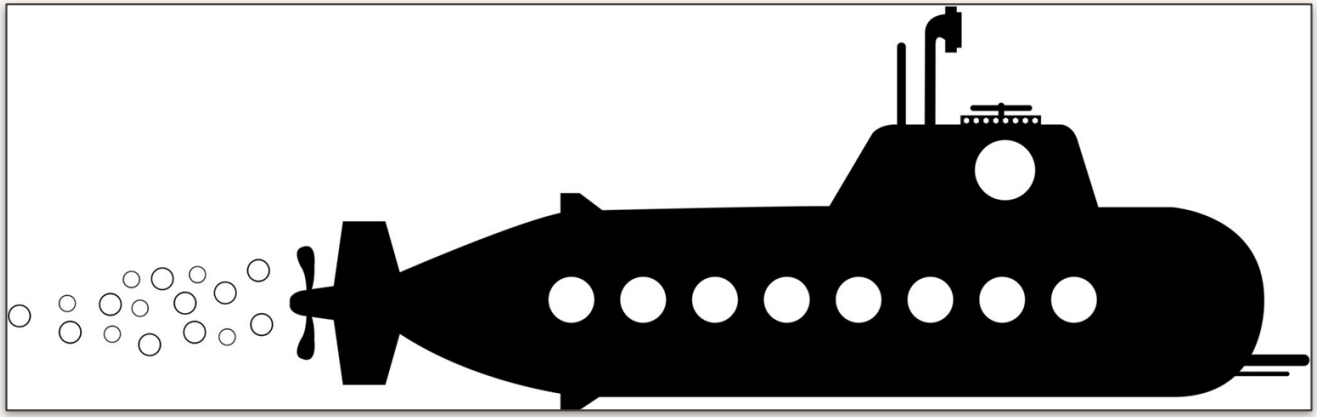
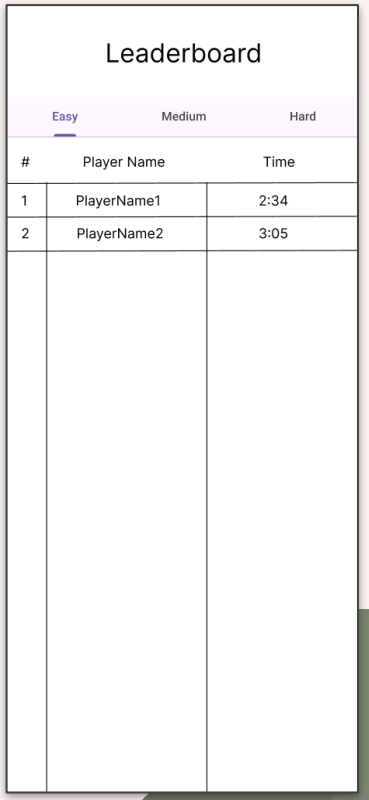


Image Source: Getty Images https://www.gettyimages.ca/detail/illustration/submarine-royalty-free-illustration/77631906

Online Leaderboard

- Users will be able to see the top ranked players
- Users will be able to see how they rank among others
- Leaderboard ranking will be based on least time to solve the maze



Mock leaderboard view

Section Two

App Demonstration & Implementation

Section One: Pitch Refinement

- App Theming & Name
- Online Leaderboard

Section Two: App Demonstration & Implementation

- Accessibility Features: Vibration and Sound
- Manual Controls using Motion Events
- Inertial Navigation using Sensors
- Rendering using Open GL ES
- Orientation Field of View Change
- Leaderboard using Firebase
- User Interface Views
- Model View Viewmodel for App Data

Section Three: Show and Tell 2

- Goals

Accessibility Features: Vibration and Sound

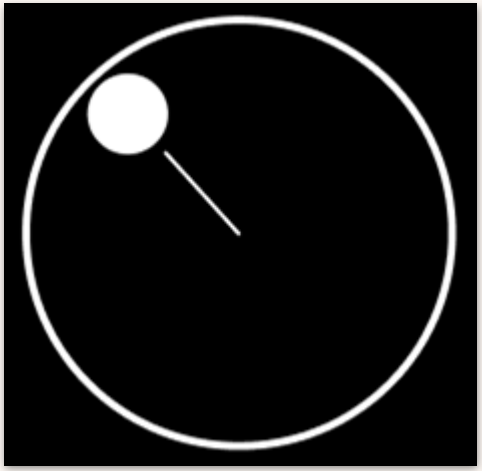
- Vibration will be implemented using the Vibration Manager and Vibrator classes
- Sound will be implemented using the Media Player class

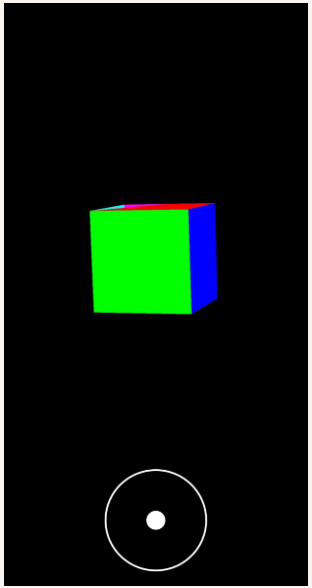
Behaviour of sound and vibration as a function of distance

Manual Controls using Motion Events

- Will be implemented using the Motion Event class
- Users will interact with a virtual joystick

Mock virtual joystick dragged up and left

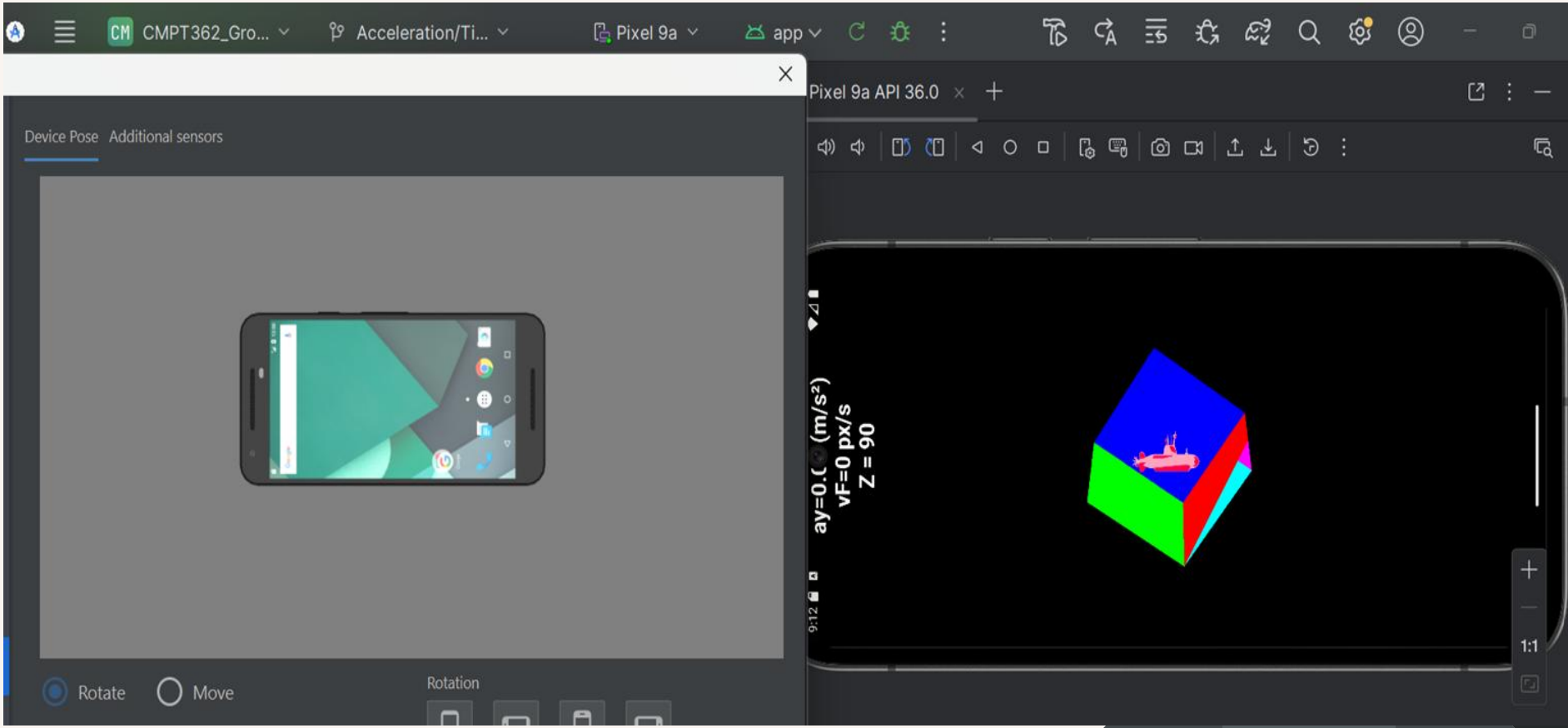


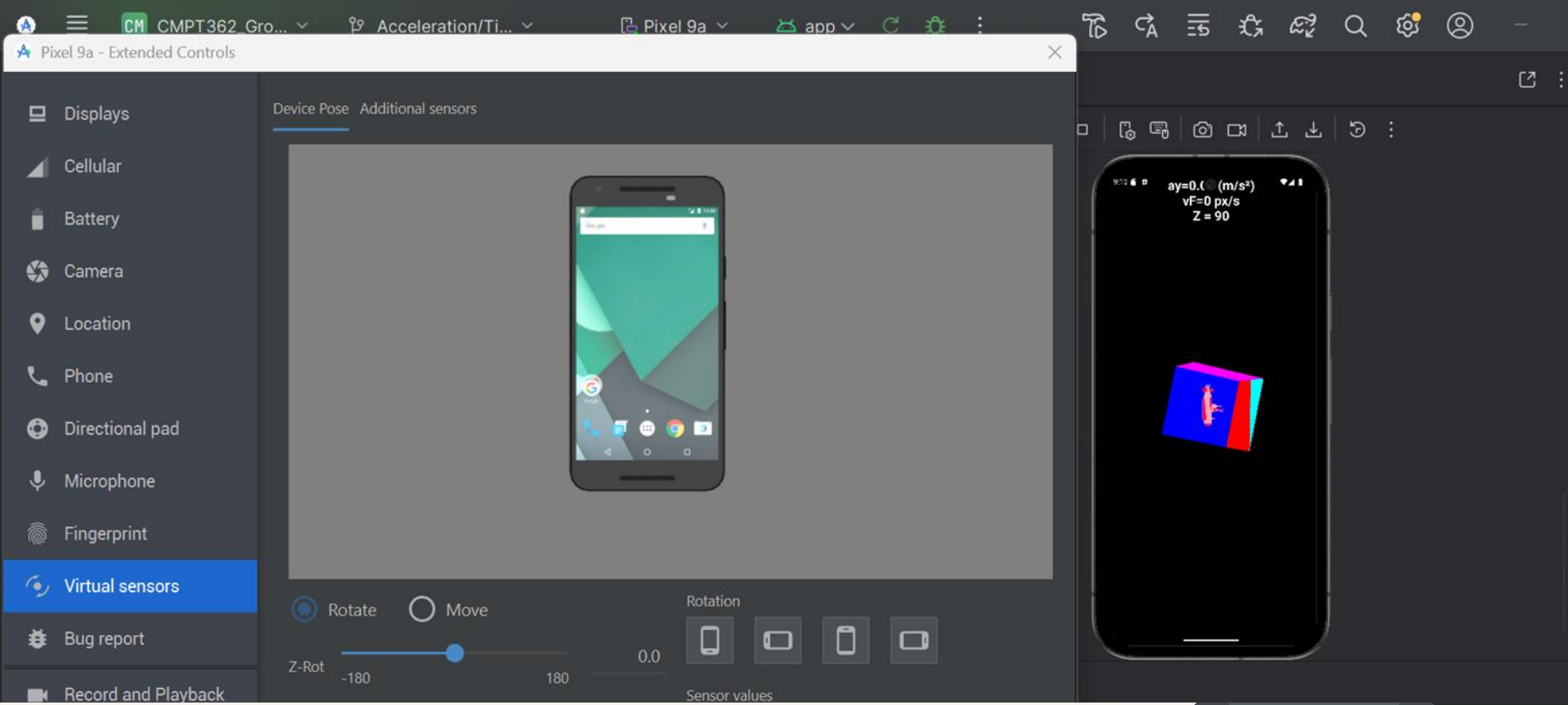


Mock virtual joystick at neutral position in portrait mode

Inertial Navigation using Sensors

- Implemented motion controls using the phone's accelerometer and gyroscope sensors
- The gyroscope measures the device's rotation and determines the submarine's facing direction by snapping Z-Rot to the nearest 90°
- The accelerometer detects physical movement along the Y-axis to move the submarine forward or backward
- Added filtering to reduce sensor noise and drag to make motion appear smoother and more natural
- Combined rotation and acceleration data to update the submarine's position and heading in real time
- The game loop continuously processes sensor input, translating real-world phone movement into on-screen navigation
- Submarine asset created by Kz — Underwater Vector Sprites (Itch.io).
- Available at: https://kz.itch.io/underwater-vector-sprites


CMPT362_Gro...
Acceleration/Ti...
Pixel 9a
app
Device Pose
Additional sensors
Rotate
Move
Rotation
Pixel 9a API 36.0
ay=0.((m/s²)
vF=0 px/s
Z = 90
1:1


CMPT362_Gro...
Acceleration/Ti...
Pixel 9a
app
Pixel 9a - Extended Controls
Displays
Cellular
Battery
Camera
Location
Phone
Directional pad
Microphone
Fingerprint
Virtual sensors
Bug report
Record and Playback
Device Pose
Additional sensors
Rotate
Move
Z-Rot
-180
180
0.0
Rotation
Sensor values
ay=0.((m/s²)
vF=0 px/s
Z = 90

Rendering using Open GL ES

- Game content is contained within a GLSurfaceView object
- Rendering logic is contained within a GLSurfaceView Renderer object

- Meshes define objects to be rendered
- Vertex and Fragment shaders define how meshes are rendered

Example shaders

```
// Define shaders
val vertShaderCode =
    "attribute vec4 atr_pos;" +
    "attribute vec4 atr_col;" +
    "uniform mat4 view_Matrix;"+
    "uniform mat4 projection_Matrix;"+
    "uniform mat4 model_Matrix;"+
    "varying vec4 v_col;"+
    "void main() {" +
    "   v_col = atr_col;"+
    "   gl_Position = (projection_Matrix * view_Matrix * model_Matrix) * atr_pos;" +
    "}"
val fragShaderCode =
    "precision mediump float;" +
    "varying vec4 v_col;"+
    "void main() {" +
    "  gl_FragColor = v_col;" +
    "}"
```

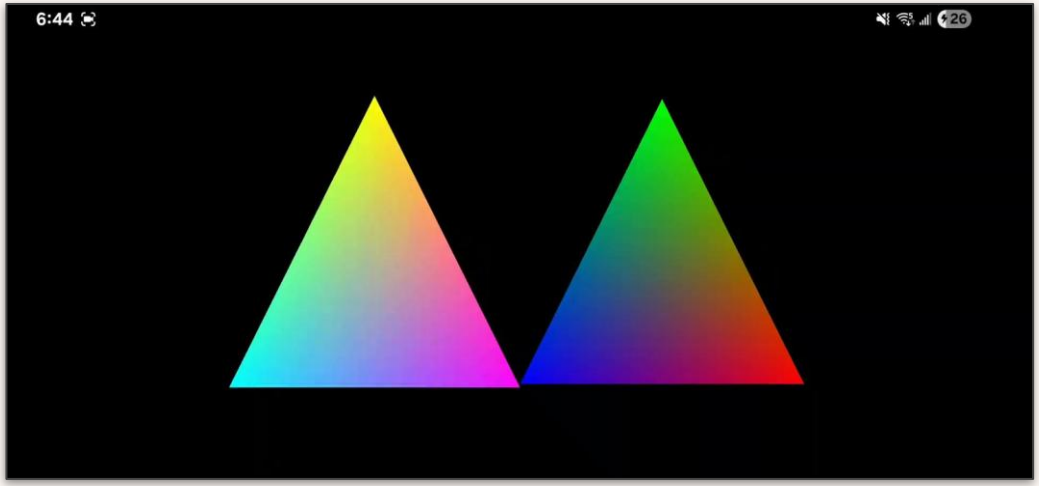
Vertex definitions for primitive meshes

```
// Defines a basic triangle's vertices
val triangleVerts = floatArrayOf(
    -1.0f,-1.0f,0.0f, 1.0f,-1.0f,0.0f, 0.0f,1.0f,0.0f)
// Defines a basic triangle's faces
val triangleDrawOrder = shortArrayOf(0,1,2)

// Defines a basic square's vertices
val squareVerts = floatArrayOf(
    -1.0f,1.0f,0.0f, -1.0f,-1.0f,0.0f, 1.0f,-1.0f,0.0f,
    1.0f,1.0f,0.0f)
// Defines a basic square's faces
val squareDrawOrder = shortArrayOf(0,1,2, 0,2,3)

// Defines a basic cube's vertices (Note: this model does not 100% reuse
// overlapping vertices)
val cubeVerts = floatArrayOf(
-1.0f,-1.0f,1.0f, 1.0f,-1.0f,1.0f, -1.0f,1.0f,1.0f, 1.0f,1.0f,1.0f,
-1.0f,-1.0f,-1.0f, -1.0f,-1.0f,1.0f, -1.0f,1.0f,-1.0f, -1.0f,1.0f,1.0f,
-1.0f,1.0f,1.0f, 1.0f,1.0f,1.0f, -1.0f,1.0f,-1.0f, 1.0f,1.0f,-1.0f,
1.0f,-1.0f,1.0f, -1.0f,-1.0f,1.0f, 1.0f,-1.0f,-1.0f, -1.0f,-1.0f,-1.0f,
1.0f,-1.0f,1.0f, 1.0f,-1.0f,-1.0f, 1.0f,1.0f,.0f, 1.0f,1.0f,-1.0f,
1.0f,-1.0f,-1.0f, -1.0f,-1.0f,-1.0f, 1.0f,1.0f,-1.0f, -1.0f,1.0f,-1.0f)
// Defines a basic cube's faces
val cubeDrawOrder = shortArrayOf(
0,1,2, 1,3,2,
4,5,6, 5,7,6,
8,9,10, 9,11,10,
12,13,14, 13,15,14,
16,17,18, 17,19,18,
20,21,22, 21,23,22)
```

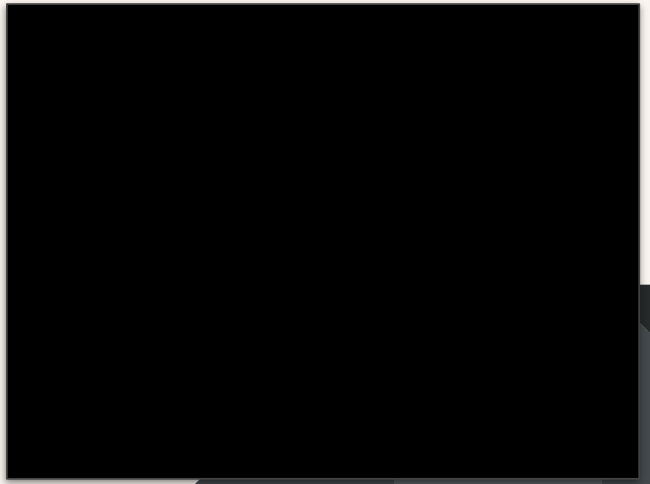
Rendering using Open GL ES Continued

- Objects are rendered in a perspective projection
- Normals for 3D objects not fully implemented

Left triangle oscillates between being close to camera and far away, showing perspective projection.

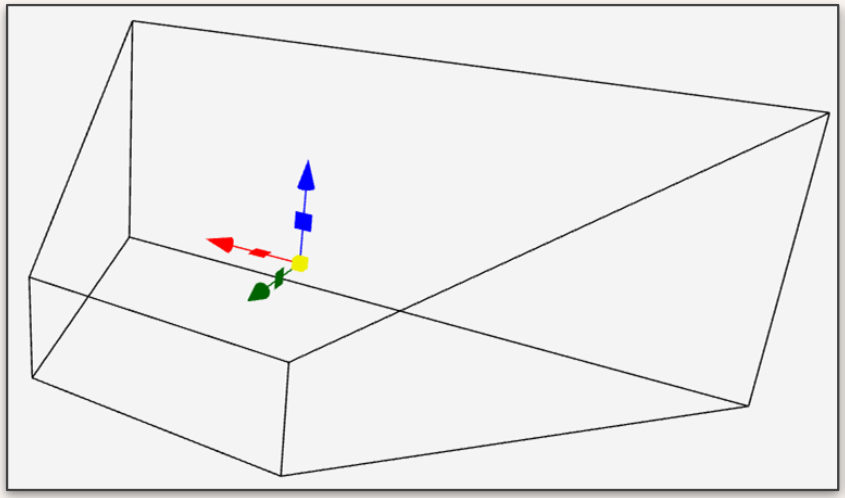


Partially implemented 3D mesh support

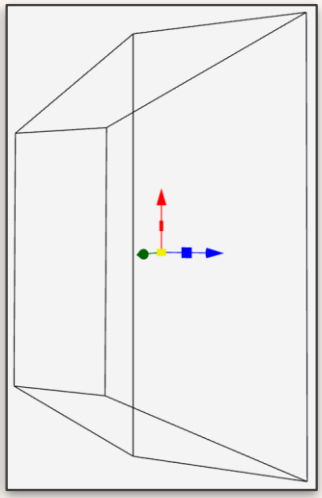


Orientation Field of View Change

- What is rendered to the GLSurfaceView is determined by the projection matrix
- In perspective projections, the projection matrix is defined by a frustum



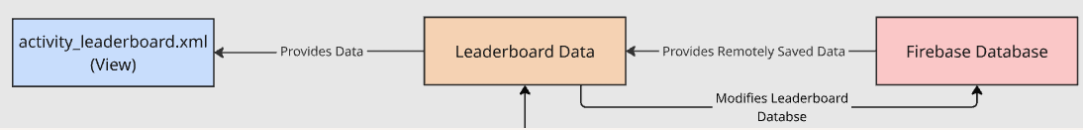
Landscape frustum



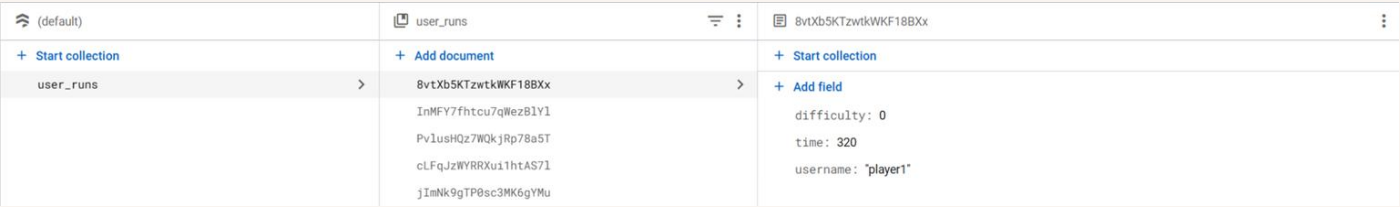
Portrait frustum

Leaderboard using Firebase

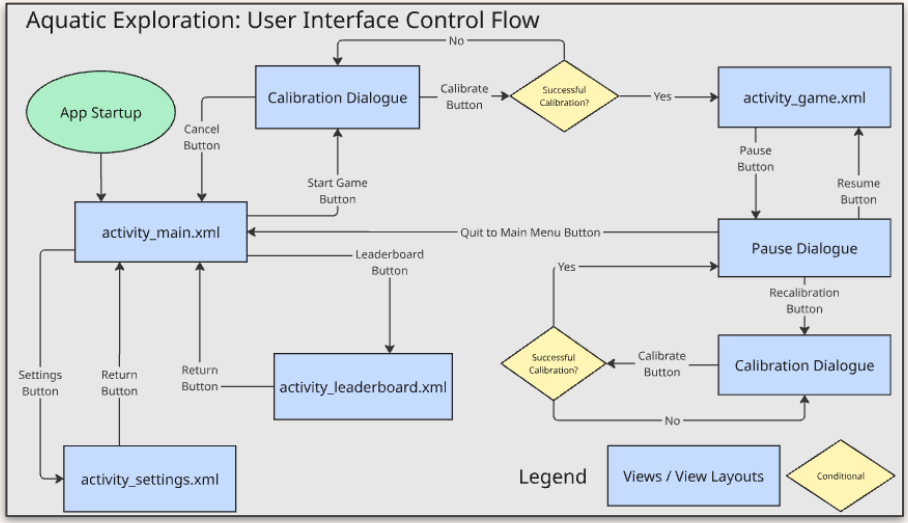
Leaderboard and database portion of MVVM diagram:

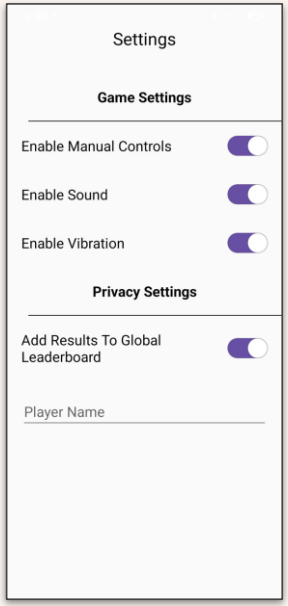


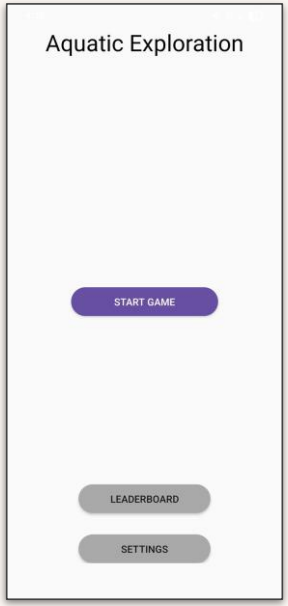
Example contents of firestore database:

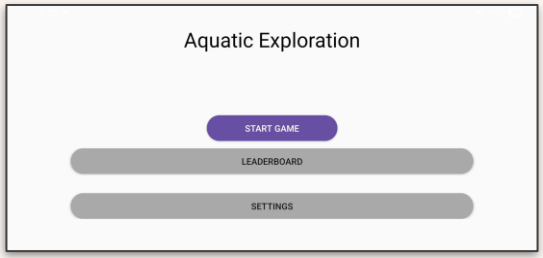


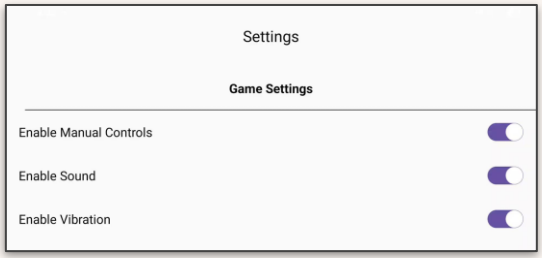
User Interface Views





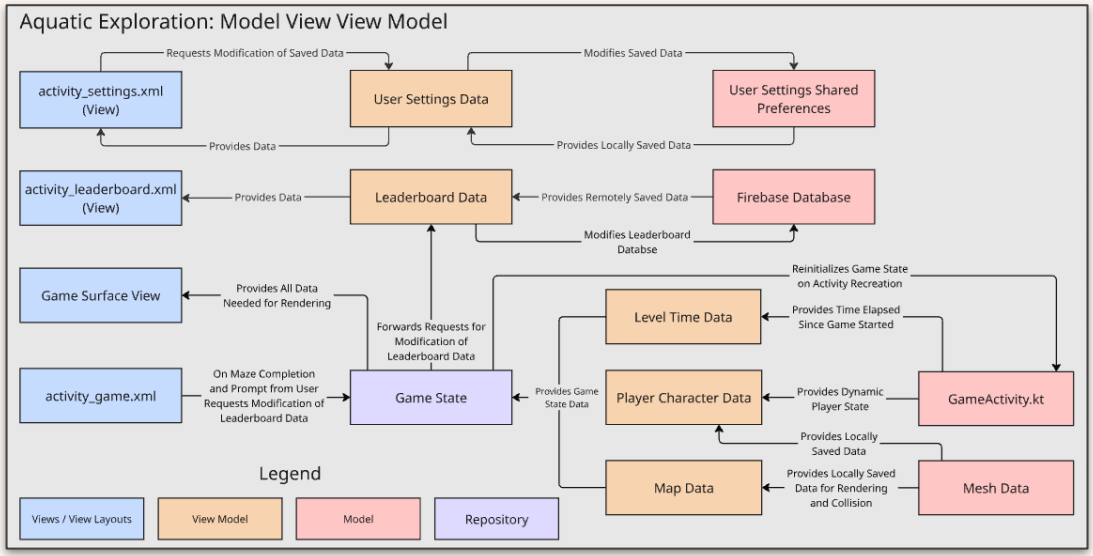






Model View Viewmodel for App Data

- Game state will be stored in a repository which contains all the data the game will need
- Data for the game will come from local storage, the game engine, or the remote database



Aquatic Exploration

Show and Tell 2
Group 30

- Updates Since Show and Tell 1
- App Demo
- Threa Design Diagram
- Who is doing what
- What we learned
- Goals

Updates Since Show and Tell 1

- Implemented Settings Menu and Linked it to the Game logic
- Added a Sound Component such that when the sub is near walls there is a rapid beep : https://youtube.com/shorts/uPiVbq5o9W4
- Added a timer component
- Added a pause option in the game where users can choose to quit or resume
- Created two mazes on figma as well as a submarine sprite
- Added acceleration which syncs with accelerometer readings
- Added tilt snaps which matches tilt rotation of sub

App Demo Manual Controls

- https://youtube.com/shorts/JRXXP4B6gZ4

App Demo Interactive

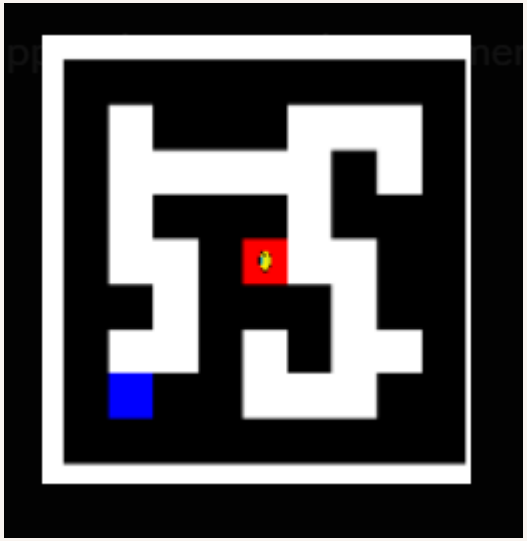
- https://youtube.com/shorts/Vsc2wc_uYiA

App Non-Game Functionalities

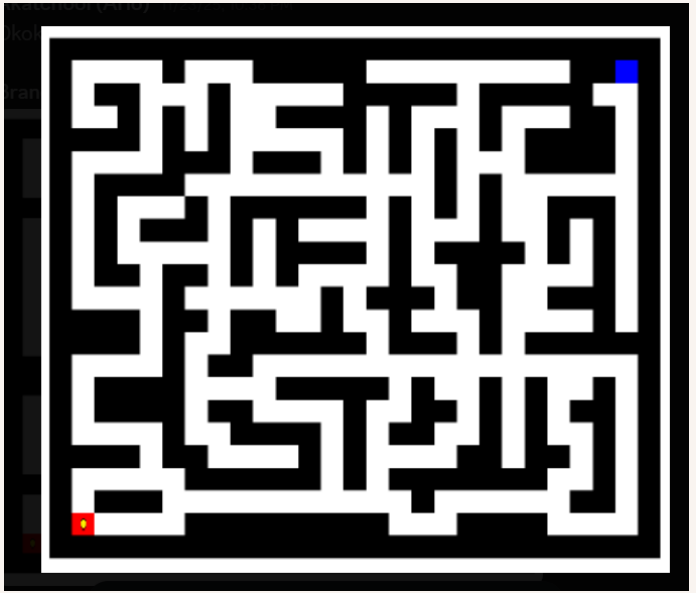
- https://youtube.com/shorts/eZsa4rYSPD4

Maps

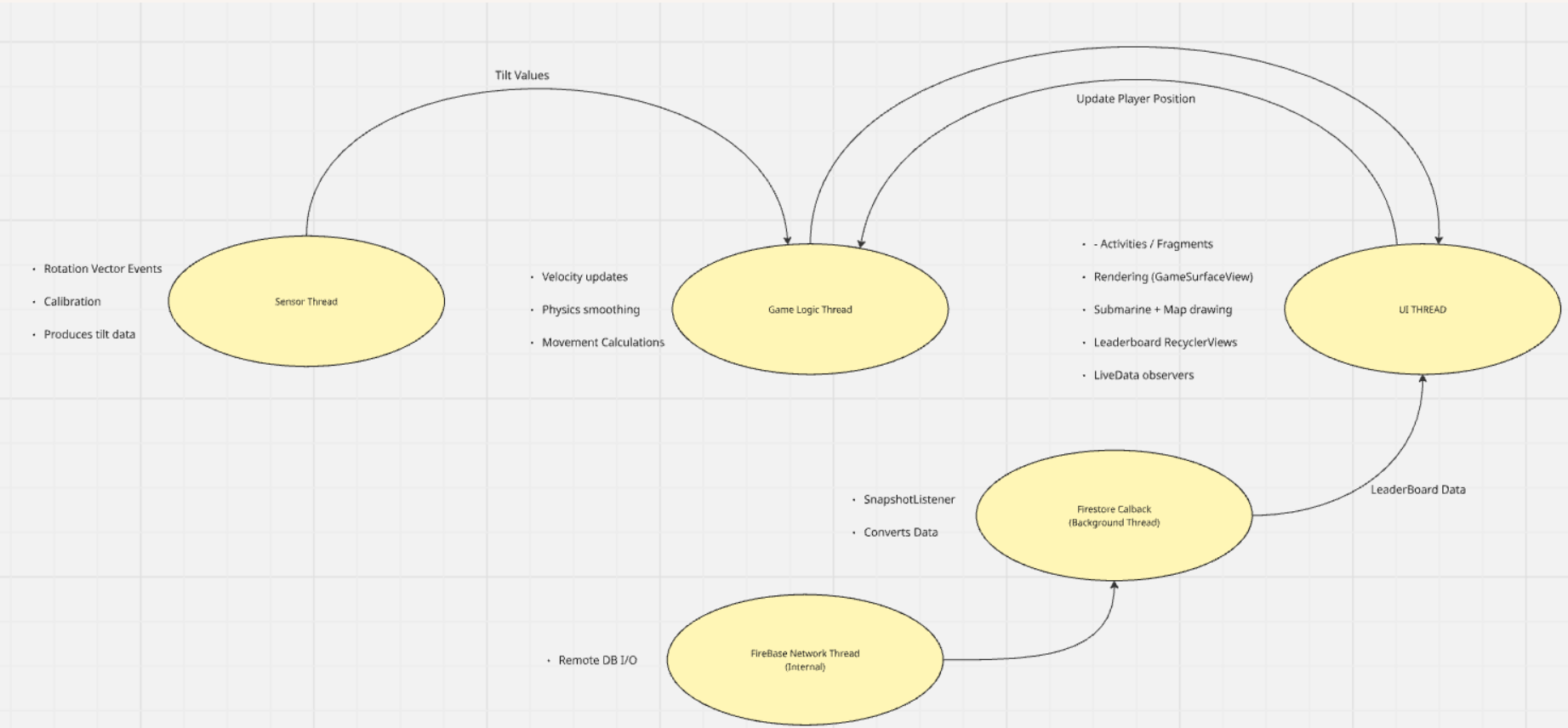
- Easy



- Hard



Thread Diagram



Who Has Worked on What

- Branden Nero: Collision Detection, Manual Controls, Open-Gl Rendering
- Cameron Lee: Database Integration, LeaderBoard
- Jonathan Osuji: Settings, App UI
- Ario Katchooi: Website, Inertial Controls (tilt and acceleration), Linking Settings to Game

Goals

- Implement all UI elements and their functionality
  - Improve UI visual design
  - Create Medium Level Map
- Implement core game logic
  - Manual controls
  - Inertial controls

Thank You For Listening!