- CS Project Semester Result -

Rhythm Game with Hand Tracker

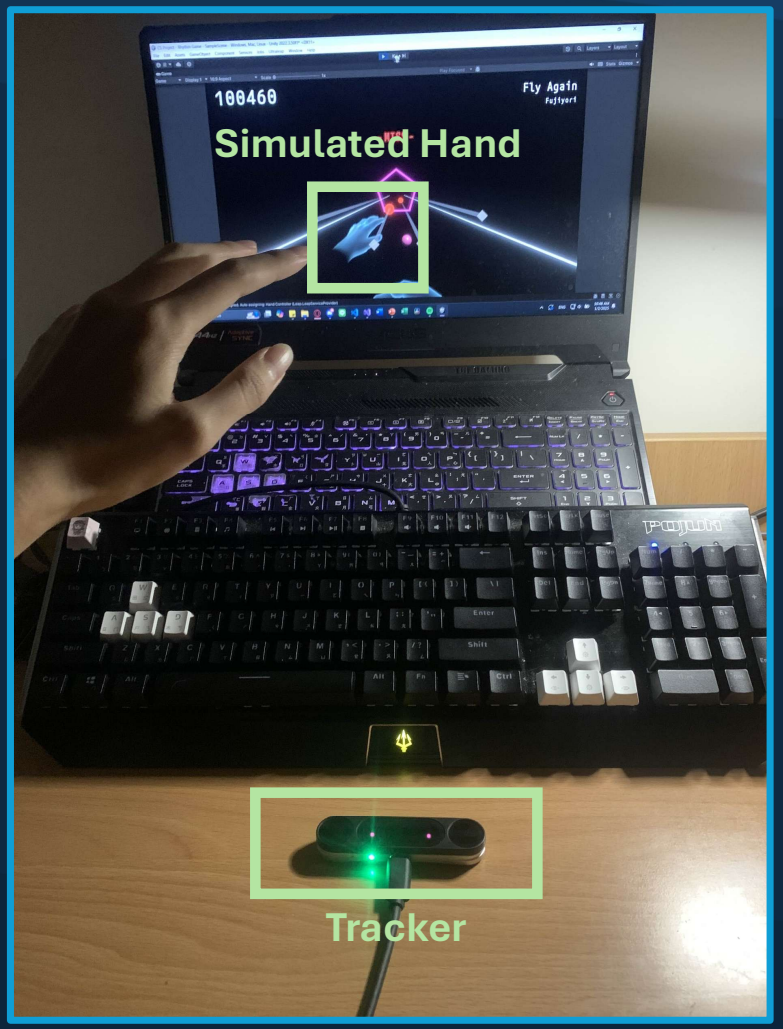
111550037 嚴偉哲

Introduction

- Rhythm game is a music-based game that challenges the player's sense of rhythm
- This project uses Leap Motion hand tracker as the controller

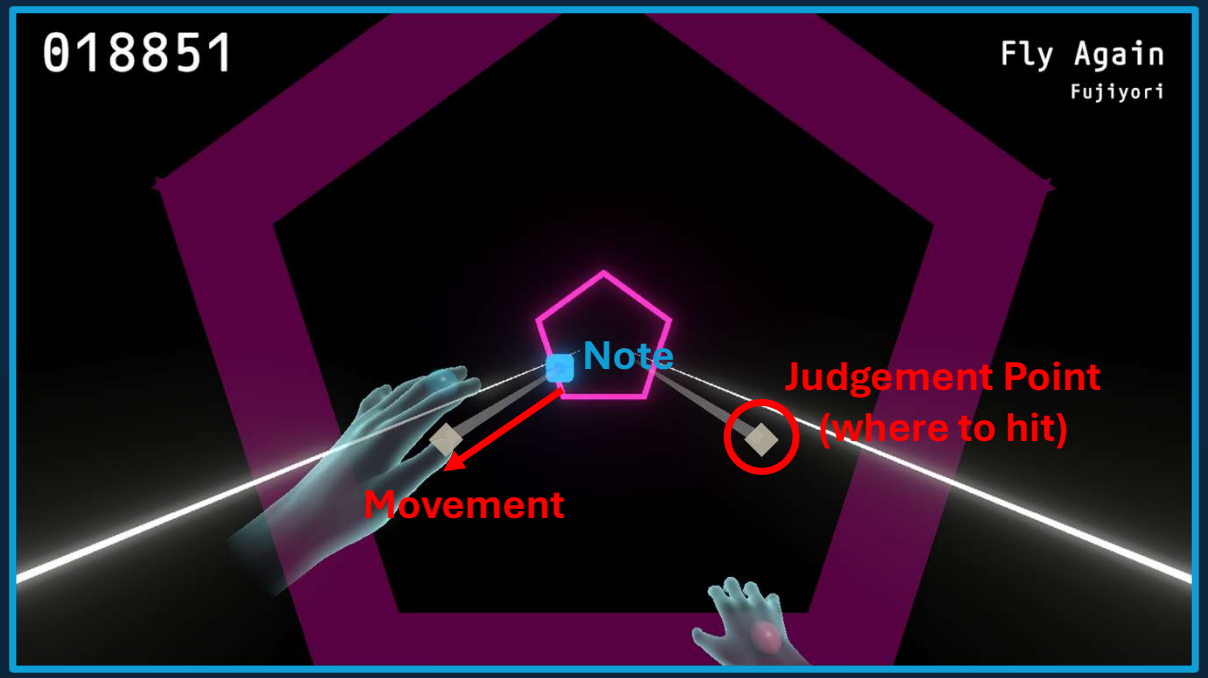
Gameplay

- The tracker is placed on the table
- A simulated hand is displayed in the screen



Gameplay

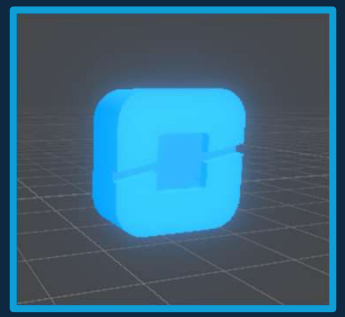
- The notes move towards the player
- Player needs to hit the note at the judgement point



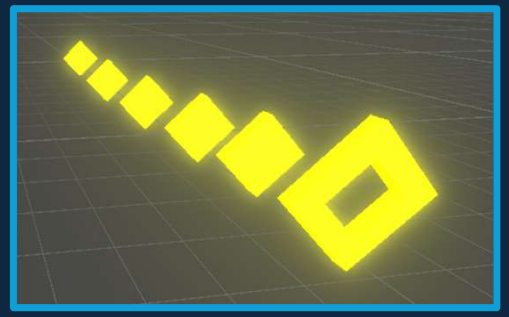
Gameplay

- 5 note types are designed

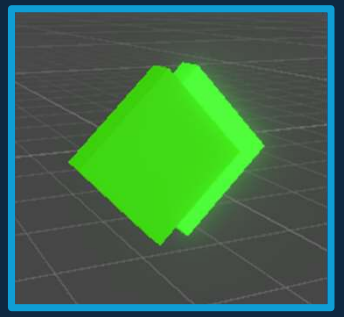
TAP



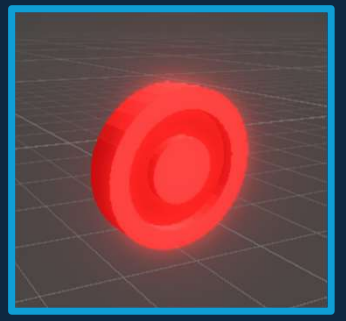
TRACK



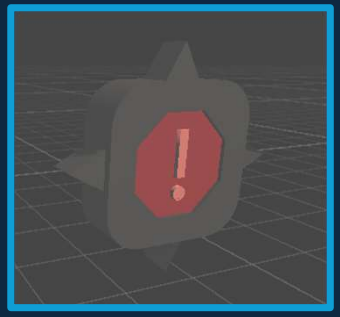
CLAP



PUNCH

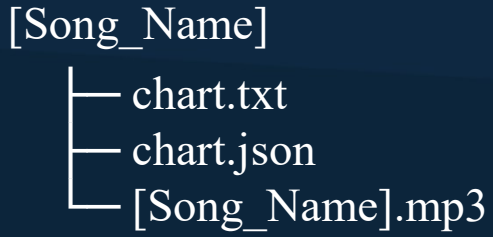


AVOID



Implementation – Chart

- Two files (notes & nodes) to store the level
- The notes (along with song information) are stored in formatted txt file
- The nodes are stored in JSON file



Implementation – Chart

- chart.txt has the timing of the notes and their assigned lane ID (which path they will follow)

Header

Chart

```
SONG: Fly Again
ARTIST: Fujiyori
OFFSET: 1250
BPM: 105
DIFF: 1
=========================
* Using Asterisk To Make Comments *
0 / 1, 0, 0, 0, 1, 0, 0, 0
# ← Section Separation Mark
* 8 *
0 / 0, 0, 0, 0, 0, 0, 0, 0
1 / 1, 0, 0, 0, 1, 0, 0, 0
#
* 16 *
0 / 1, 0, 0, 0, 1, 0, 0, 0
1 / 0, 0, 1, 0, 0, 0, 1, 0
```

Lane ID

```
#
* 72 *  Each underline is one beat
0 / 0, 0, 00, 0, 0, 0, 00, 0
1 / 2, 0, 00, -, 0, 0, 01, 0
4 / 0, 0, 01, 0, 0, 0, 10, 0
5 / 0, 0, 10, 0, x, x, 00, 1
6 / x, x, 00, 1, 2, 0, 00, -
#
```

Implementation – Chart

- chart.json has the control nodes of the lanes

One Node

```
{
    "Lane": 3,
    "Beat": 47,
    "Position": [-13, -8],
    "PosEase": "quintInOut"
},
{
    "Lane": 3,
    "Beat": 49,
    "Position": [-13, 6]
},
```

```
public class PropertyNode
{
    // required
    public int Lane = -1;
    public float Beat = int.MinValue;

    // initial
    public float[] Position = { int.MinValue, int.MinValue }; // for parsing
    public Vector2 Pos = new Vector2(int.MinValue, int.MinValue); // for accessing

    // optional
    public float Alpha = -1;
    public float Speed;
    public string PosEase = "easeInOut";
    public string AlphaEase = "linear";
    public string Endpoint = "none";

}
```

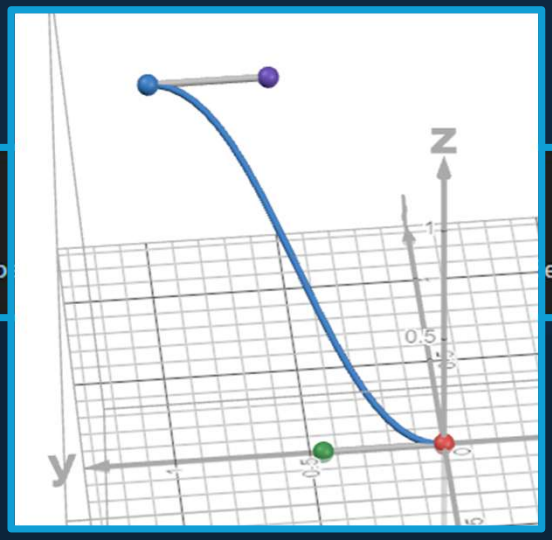
Implementation – Level Generation

- Each lane is drawn by the line renderer built in Unity

- The nodes are converted into Bezier curves and then sample the points

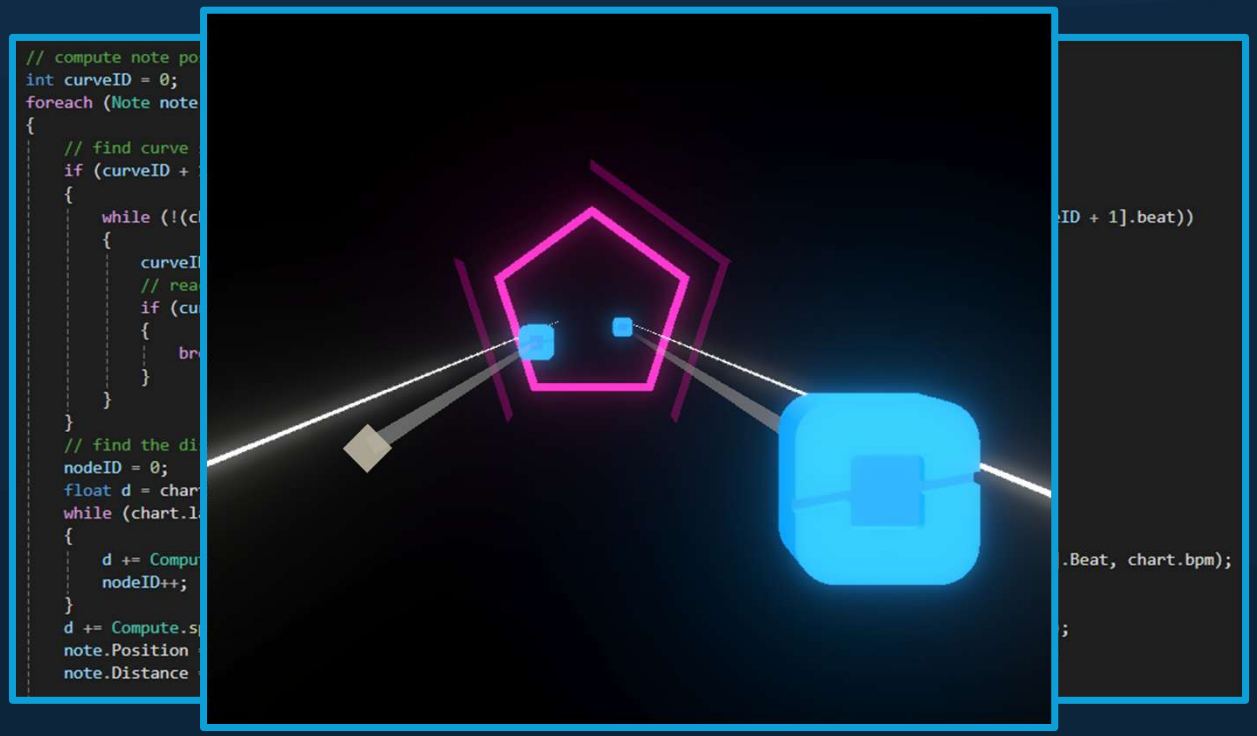
```
case "easeInOut":
    startTangent = new Vector3(0, 0, 0.45f);
    endTangent = new Vector3(1, 1, 0.55f);
    chart.lanes[i].curves.Add(new Curve(headNodePos, startTangent, tailNodePo
    break;
```

eat, distance));



Implementation – Level Generation

- The notes are placed onto the lanes afterwards



Implementation – Control

- The tracker has a Unity package that stores hands' data
- The data is first preprocessed for conveniency

Implementation – Control

- Tap, clap, and punch notes have precedence check
- The frontmost note is judged when multiple notes are hit simultaneously by one hand

```
// find the note hit by the left hand (if any)
for (int i = 0; i < detectedTapNotes.Count; i++)
{
    if (detectedTapNotes[i].leftDetect)
    {
        leftHit.Add(i);
    }
}
if (leftHit.Count > 0)
{
    // find first
    int index = 0;
    for (int i = 1; i < leftHit.Count; i++)
    {
        if (detectedTapNotes[leftHit[i]].beat < detectedTapNotes[leftHit[index]].beat)
        {
            index = i;
        }
    }
    // first touched is marked as hit
    if (detectedTapNotes[leftHit[index]].hitBy == LEFT)
    {
        Judge(detectedTapNotes[leftHit[index]], true);
    }
}
```

Implementation – Control

- A clap is registered if
  1) The positions of two hands are within a distance
  2) The normal vectors face each other

```
// Detect clapping, the following 3 must be satisfied:
// 1. The palm positions of 2 hands needs to be within a distance
// 2. The normal vectors of 2 hands needs to face each other
if (lHand != null && rHand != null)
{
    if (Vector3.Dot(lHand.PalmNormal, rHand.PalmNormal) <= -0.8f)
    {
        float lDot = Vector3.Dot(lHand.PalmNormal, lHand.PalmPosition);
        float rDot = Vector3.Dot(lHand.PalmNormal, rHand.PalmPosition);
        // check the normal direction distance
        if (Mathf.Abs(lDot - rDot) <= 0.08f)
        {
            Vector3 projRHandPos = rHand.PalmPosition - lHand.PalmNormal * (lDot - rDot);
            // check the projected-to-plane distance
            if ((lHand.PalmPosition - projRHandPos).magnitude <= 0.08f)
            {
                if (!isClapping)
                {
                    isClapping = true;
                    clapPos = Compute.transformHandPos((lHand.PalmPosition + rHand.PalmPosition) / 2);

                    // add to trail
                    clapPosTrail.Add(new HandPosRecord(clapPos, time));

                    // debug
                    //Debug.Log("Clap at " + clapPos + "!");
                }
            }
```

Implementation – Control

- A punch is registered if
  1) The hand is holding fist
  2) The hand is pushing forward in +z direction

```
// Detect punching, the following 2 are required:
// 1. Player needs to hold their fist. This is implemented by checking if four fingers are pointing towards palm and the thumb not extending.
//      # Although there's a provided fist detection in the package, the stability of it is low.
// 2. The hand needs to push forward (+z direction)
//
// We take the center point of the proximal(first) bone of the middle finger as the punching point
if (lHand != null)
{
    if (lHand.PalmVelocity.z >= 0.7f)
    {
        if (checkHoldFist(Chirality.Left) && !lPunching)
        {
            lPunchPos = Compute.transformHandPos(lHand.fingers[2].bones[2].Center);
        }
    }
    else
    {
        lPunching = false;
    }
}
```

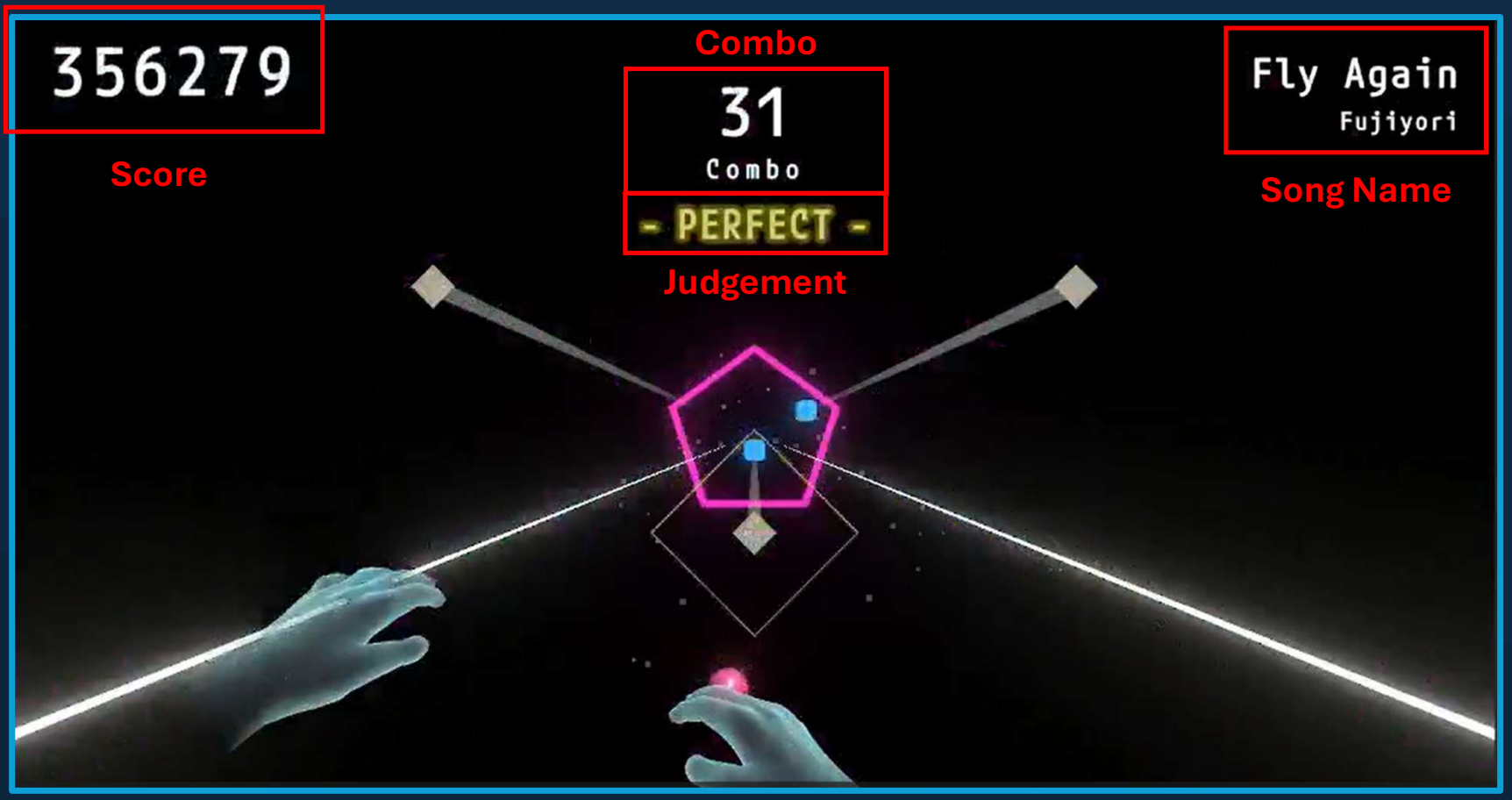
Implementation – Judgement

- The performance of the player is tracked throughout the level

- The score is defined as:

$$100{,}000 \times \frac{[maxCombo]}{[totalNotes]} + 900{,}000 \times \frac{[perfectHit] + [goodHit] \times 0.6}{[totalNotes]}$$

Implementation – UI



Problems

- The tracker sometimes cannot detect the hand due to obstruction or outside of angle of view
- Charting the nodes information can be complicating when the number of lanes and length increase

Future Plan

- Expand to VR and the notes can be coming from 360
- Train machine learning model for chart generation or optimal hit pattern (i.e., which hand to hit which note)