Volley Spikers

Video Link：
https://youtu.be/2-CQYsCYydY

A motion-sensing, multiplayer volleyball game using mobile phones as the input.

Group Members

Terry (Tairui Li): Programming | Technical Art

Jasmine：Scene building | UI design

Johnson：Planning | 2D animation

Made With

Rider

Unity

Blender

aseprite

Final Output

Distinguishing Features

- **Innovative** - Players control characters on the computer by **moving their phones**.
- **Interesting** - Multiplayer gameplay allows players to **interact** with each others.
- **Relaxing** - The **stylized art** and the setup at beach creates a relaxing atmosphere.

Future Plan (Implemented)

- Add a mode that allows players controlling the character with the keyboard.
- Add AI controlled opponents, so single player mode would be possible.

Existing Problems

- Takes quite a while for a beginner to learn how to control.
- Needs two players and two Android devices to play.

Inspiration

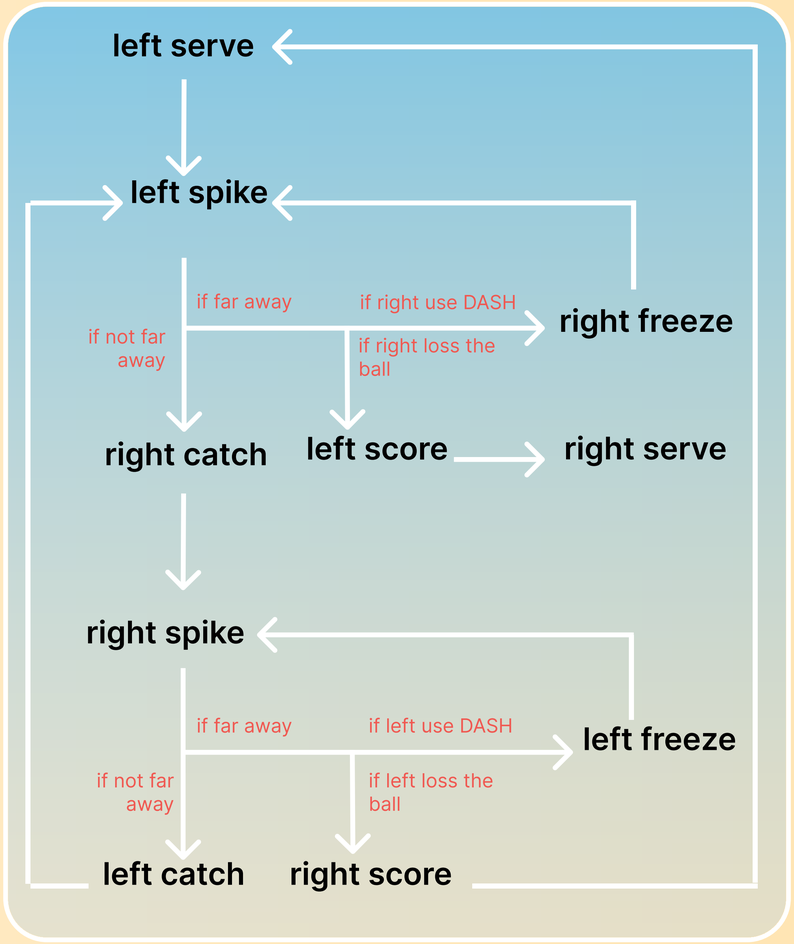
During the summer school, we came up with many themes in hopes of making an innovative game. However, we realized that even with new themes, **gameplay innovation** was still **difficult**. So, we focused on **input methods**. Most mainstream games use keypads, joysticks, or touchscreens as input, while somatic games require specialized devices (e.g., wii, kinect, etc.), limiting their popularity. We decided to use **cell phone** as the **input device**, which not only innovates the input method, but also ensures the popularity by eliminating the need for players to purchase extra devices.

Reference

The gameplay is inspired by **volleyball, soccer, rattan ball and racquetball**.

The major mechanics are based on **volleyball**: players should hit the ball and try to let their opponents failed to catch the ball.

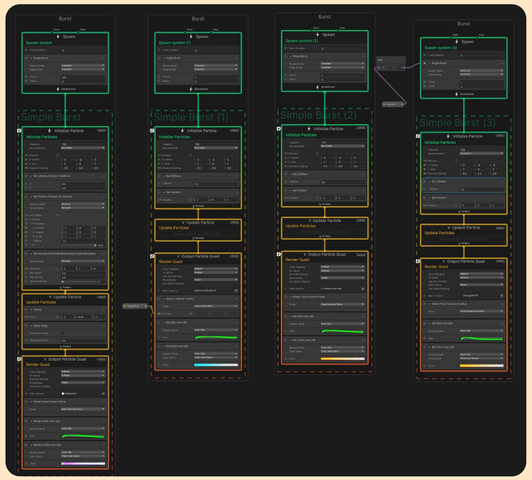
The ball in this game can be kicked, just like soccer and rattan ball. Besides, this game has walls like racquetball, though the walls are transparent.

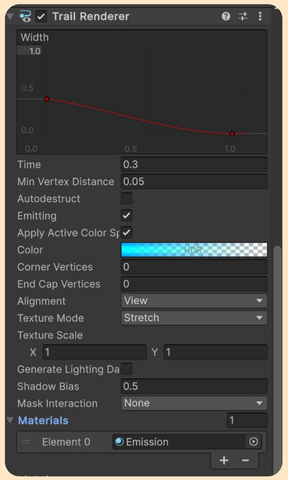


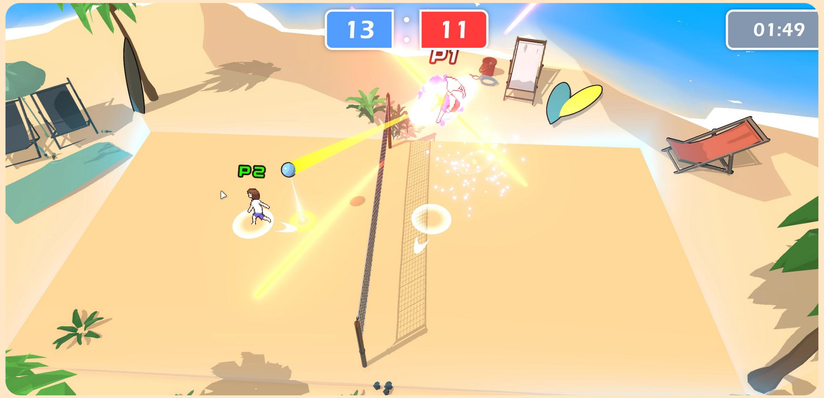
Flow Chart

Visual Effects

We used **VFX Graph** for hit/smash effect, and trail renderer for the ball's trail.

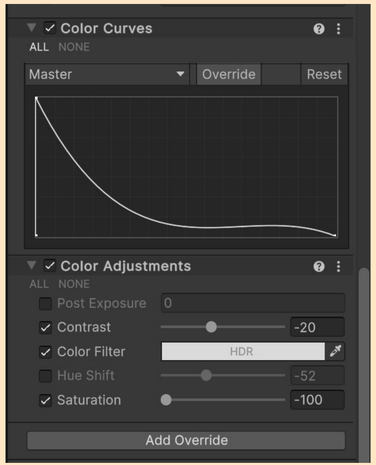


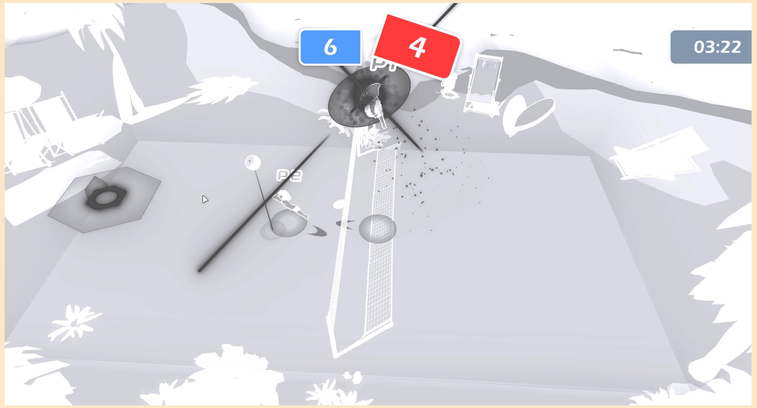




We also used some **post processing** effects to create a blink screen effect.

This effect significantly enhanced the **feedback** when a player successfully smashed a ball.





Basic operation

1

Aim the phone at the computer, and press “Calibrate Direction”

2

Hold the phone as shown in the picture, and incline the phone to move.

3

Shake your phone vertically when the circle under you changes color to kick/smash the ball.

4

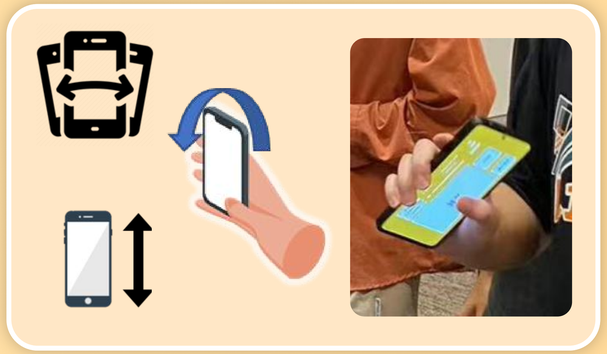
Dash

Press “Dash” to dash to the ball, but doesn’t function when you’re too far from the ball.

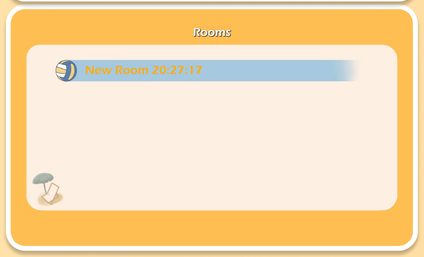
Code Implementation

```
public void ApplyRemoteMsg(ControllerInfo info){
    timeSinceLastRcv = 0;
    if (!isActiveAndEnabled){
        SetActive(true);
        Message.Create(LocalizeTxt($params new: $"{num.text} Reconnected", $"{num.text} 重新连接了"));
    }

    Quaternion q = Quaternion.Euler(0, 90, 0);
    if (side != info.side){
        side = info.side;
        transform.position = (int)side == 0 ? rightBirthPoint : leftBirthPoint;
    }

    var a = transform.eulerAngles;
    a.y += 180;
    transform.rotation = Quaternion.Euler(0, 180, 60);
    Quaternion finalRotation = info.angleDelta * q * info.angle;
    Vector3 dir = finalRotation * new Vector3(-1, 0, 0);

    Vector3 velocityH = new Vector3(dir.x, 0, dir.z);

    rb.velocity = new Vector3(0, rb.velocity.y, 0) + velocityH * speed;
    bool b = velocityH.magnitude > 0.5f;
    if (b != running){
        running = b;
        animator.SetBool(runningBoolID, b);
    }

    var horizontalRotation = Quaternion.Euler(new Vector3(0, finalRotation.eulerAngles.y, 0));

    facingDirection = Quaternion.LookRotation(horizontalRotation * Vector3.forward);

    if (Mathf.Abs(info.acceleration.x) > 2f){
        Kick();
    }

    if (info.dashed){
        Dash();
    }

    characterSprite.flipX = !(rb.velocity.z > 0);

    if (!aniSpeedSetByOthers){
        animator.SetFloat(speedFloatID, value: rb.velocity.z / -2f);
    }
}
```

Controlling the character with phone’s gyroscope:



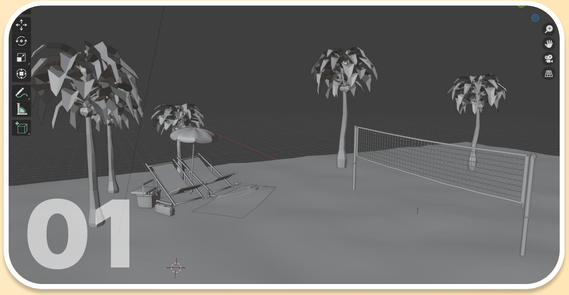
Broadcasting Room in Local Area Network:

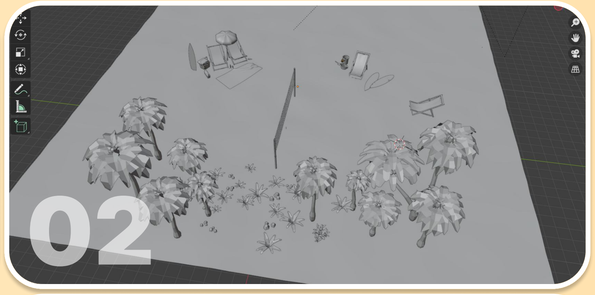



```
public async void BroadcastRoom(string roomName){
    var addresses = NetSys.GetLocalAddresses();
    for (int i = 0; i < addresses.Count; i++){
        var a = addresses[i].ToString();
        string tmp = a[..(a.LastIndexOf(".") + 1)];
        addresses[i] = IPAddress.Parse(tmp + "255");
        Debug.Log(message: addresses[i].ToString());
    }

    broadcasting = true;

    for (int i = 0; i < addresses.Count; i++){
        try{
            NetSys.SendMsg(new NetworkMsg(type: Msg.RoomBroadcast, roomName), new IPEndPoint(addresses[i], port: 8887));
            Debug.Log(message: addresses[i].ToString() + " Passed");
        }
        catch{
            addresses.RemoveAt(i);
            i--;
        }
    }

    while (broadcasting){
        await Task.Delay(100);

        foreach (var address in addresses){
            try{
                NetSys.SendMsg(new NetworkMsg(type: Msg.RoomBroadcast, roomName), new IPEndPoint(address, port: 8887));
            }
            catch (Exception e){
                Debug.Log(e.Message);
            }
        }
    }
}
```

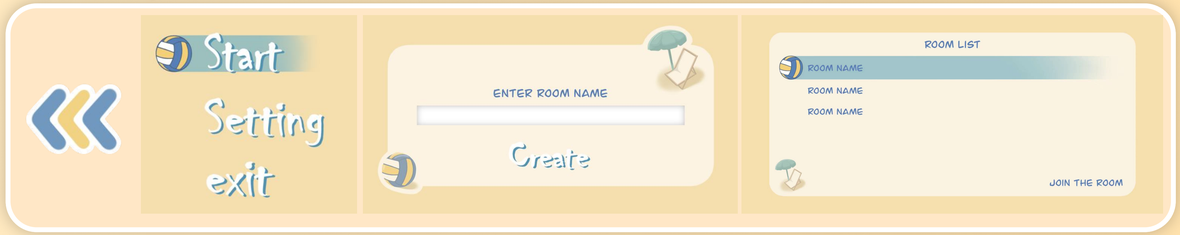
Iteration — Arts

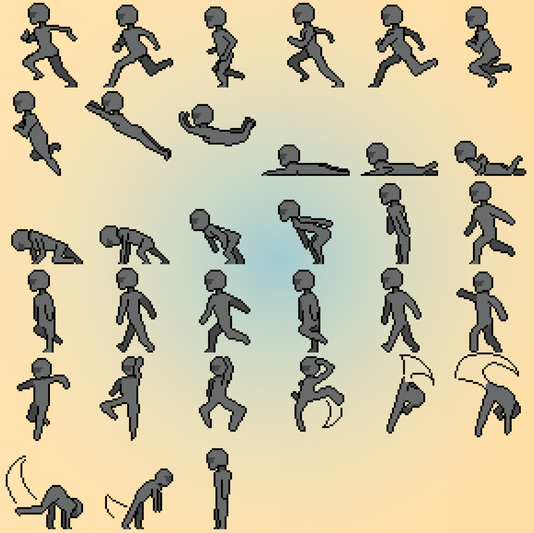
3D Art



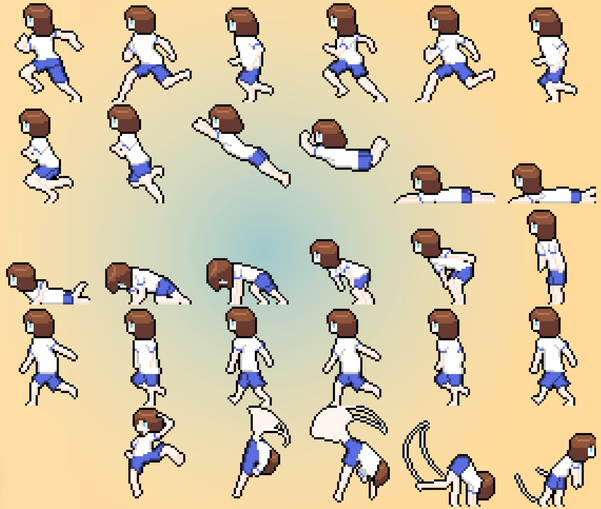


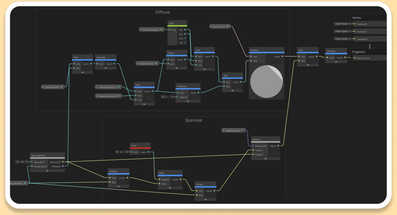
UI Design





2D Art

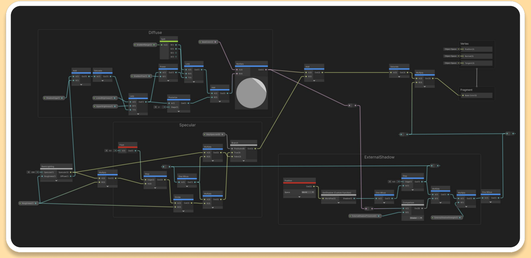




Rendering

Shader for stylized rendering

Problem Solved: cannot receive shadows.



Iteration – Mechanics

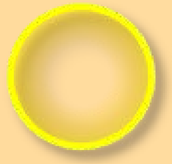
Core Mechanics

Before

Stamina: Returning a kicked ball costs stamina. A player cannot return a kicked ball if he/she runs out of stamina.
stamina regain: regain slowly while standing still

Spike: If the ball is too high, a spike circle will appear at the ball drop point, jump up in the circle to enter the bullet time, you can choose to kick by hand/foot, use the phone to control the direction of the ball (similar to a slingshot), if the opponent does not have enough stamina, then the opponent will be knocked off (In four-player mode players can rely on teammates to actively trigger the spike circle)

Spike Circle



After

Stamina is removed, so that players **won't be distracted** by stamina, reducing the difficulty for new players.

The spike circle and the camera movement when spiking the ball are canceled.
The **spike mechanics** is changed: players can spike the ball when they're under the ball and the ball is above a certain height. The direction is determined by the arrow at the bottom of the character.
This **speed up** the game and **simplified** the operation to reduce the difficulty.

Player Control

Before

One button is used for **dashing**. When a player dashes, he/she moves toward the ball for a certain distance.
Dashing will trigger the spike circle, but dashing into the spike circle will not trigger the opponent's spike circle.

The ball will be returned automatically when it is touched by a player. If the player jumps when returning the ball, the spike circle will be triggered.
The **angle between the character and the ball** determines the direction of the hit.

To spike the ball, players need to jump and kick/pick the ball **with the characters' feet.**

After

Dashing has been changed to move directly underneath the ball, and the player is **stunned** for 1 second after dashing.
This modification simplified controlling and added penalty to dashing for balance. The new mechanics of **dash** examines the player's timing of the pounce more than his skill in controlling the movement of the character, making it less difficult for new players.

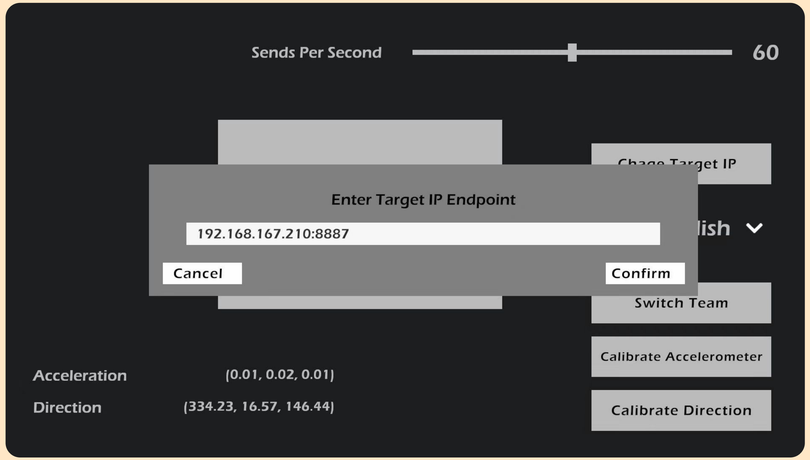
The direction of a normal hit is now determined by the **orientation of player's phones**, and the speed is determined by the player's speed.
(Simplified Operation)

We change the way of triggering foot spike to **shaking the phone up and down**, and canceled the mechanic of picking with feet.
This increases the interaction between players, and simplifies the mechanism

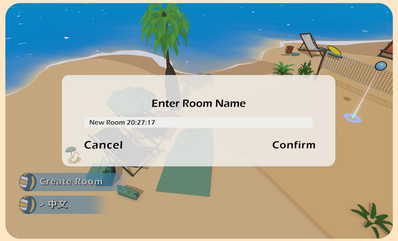
Iteration — Code

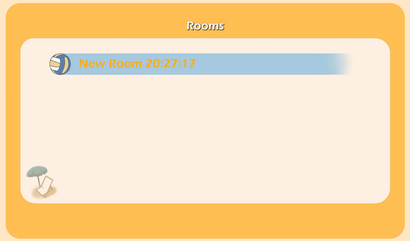
Broadcasting Rooms

Before Players have to manually enter the IP address of the computer to connect, which is sophisticated.



After Players can connect to the computer by clicking “create room” on the computer and join the room with only one click on the phone.





PC Controlled Character & Alternative Keyboard Input

Code for PC Controlled Character

```
public void Update(){
    if (kicksLeft < 1){
        kicksLeft += Time.deltaTime * 3;
    }

    c.timeSinceLastRcv = 0;

    var cPos = transform.position;
    cPos.y = 0;
    var bPos = Ball.instance.transform.position;
    bPos.y = 0;
    Character targetC = null;
    foreach (var otherC in Character.characters){
        if (otherC.gameObject.activeSelf && otherC.side != c.side){
            targetC = otherC;
        }
    }

    var opponentPos = new Vector3(0, 0, 0);
    if (targetC){
        opponentPos = targetC.transform.position;
        opponentPos.y = 0;
    }


    Vector3 velocity = bPos - cPos;
    float dist = velocity.magnitude;
    velocity += 10;

    if (Ball.instance.GetSide() != c.side || Ball.instance.rb.velocity.z > 0 == (c.side != Side.Right) && Mathf.Abs(Ball.instance.rb.velocity.z) > 10){...}

    velocity.y = 0;
    velocity = Vector3.ClampMagnitude(velocity, c.speed * 0.9f);
    velocity.y = c.rb.velocity.y;
    c.rb.velocity = velocity;

    opponentPos.x += 5 * Mathf.Sin(Time.time * (randomDelta)) * Mathf.Cos(Time.time * (1 + 0.5f * randomDelta) + 3);

    c.facingDirection = Quaternion.LookRotation(((int)c.side == 0 ? 1 : -1) * (opponentPos - cPos));

    if (Ball.instance.rb.velocity.y < 0 && (bPos.x < 1f && dist > 5 && Ball.instance.rb.velocity.magnitude > 15 || bPos.y < 0.5f && dist > 2.5f)){
        c.Dash();
    }

    if (kicksLeft > -1.3f && Mathf.Abs(bPos.z) < 3.5f){
        c.Kick();
        kicksLeft--;
    }
}
```

Code for Keyboard Input


```
public class KeyboardInput : MonoBehaviour{
    public Character c;  未更改

    public bool online;  未更改

    public Vector3 velocity;  可序列化

    事件函数
    public void Awake(){
        c = GetComponent<Character>();
    }

    事件函数
    public void Update(){...}

    经常调用  1 用法
    public void ChangeSide(){
        c.side = (int)c.side == 0 ? Side.Left : Side.Right;
        transform.position = (int)c.side == 0 ? Character.rightBirthPoint : Character.leftBirthPoint;
    }

    经常调用  1 用法
    public void UpPressed(){
        velocity += new Vector3(-1, 0, 0);
    }

    经常调用  1 用法
    public void LeftPressed(){
        velocity += new Vector3(0, 0, -1);
    }

    经常调用  1 用法
    public void DownPressed(){
        velocity += new Vector3(1, 0, 0);
    }

    经常调用  1 用法
    public void RightPressed(){
        velocity += new Vector3(0, 0, 1);
    }
}
```