GESTURES

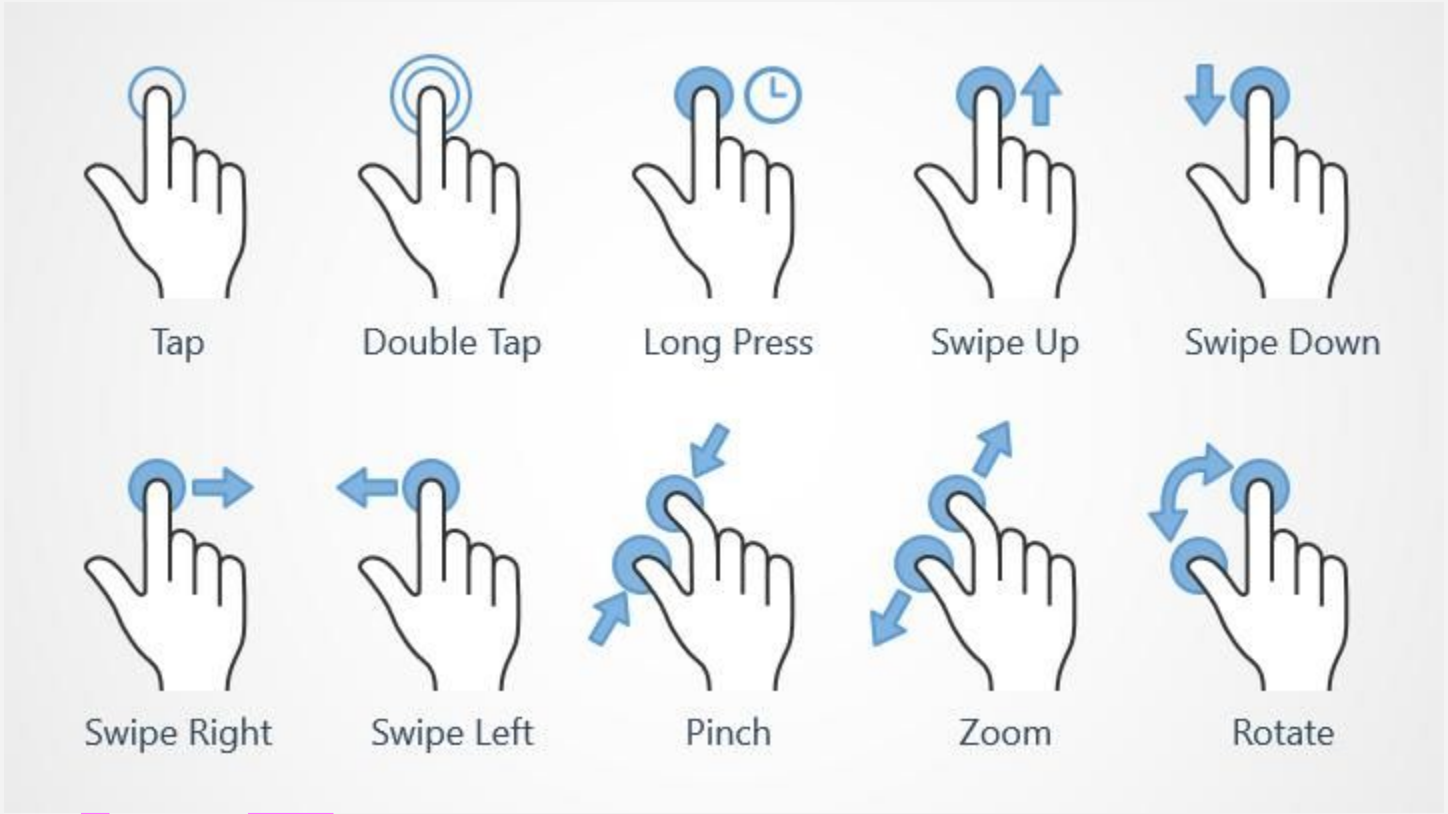
Week 6 - Mobile Game Programming
1604C262
Teknik Informatika - Universitas Surabaya

GESTURE TYPES

A gesture is a **movement of a finger**, hand, or other part of the body to express an idea, desire, or other human thought.

Initially, some devices could only process the touch of one digit at a time, but **multi-touch** devices can perceive both gestures made by one finger and those made by more than one at the same time.

GESTURE TYPES



GESTURE TIMING

How long a gesture lasts is just as important as how many fingers are involved.

How long is the difference between a drag and a flick?

How long does a tap last before it becomes a tap and hold?

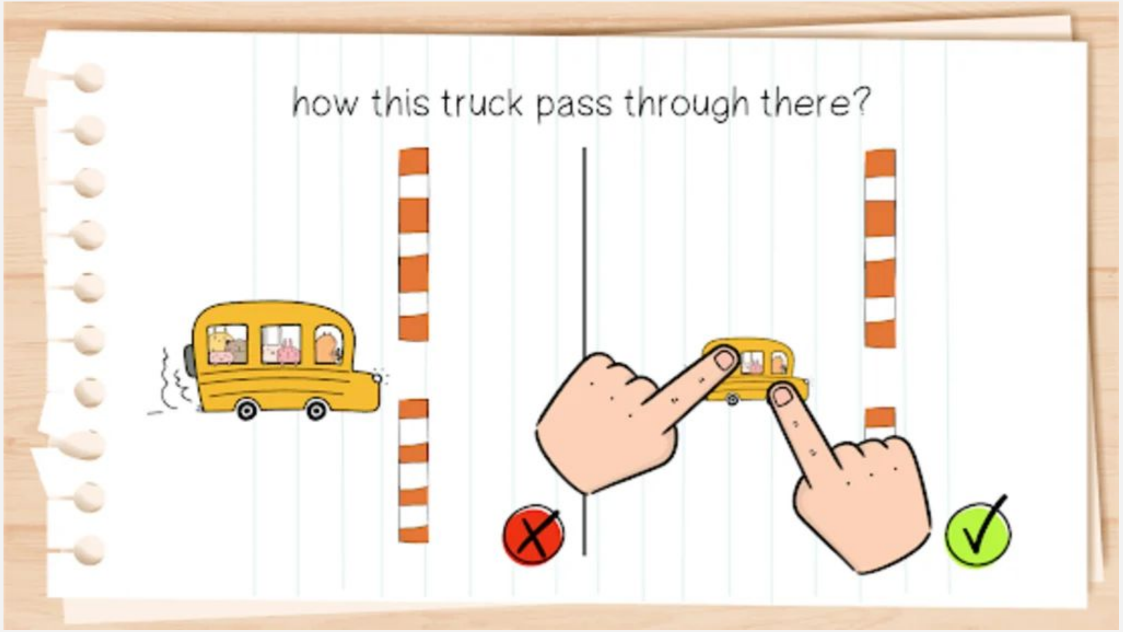
How long between taps makes a double tap?

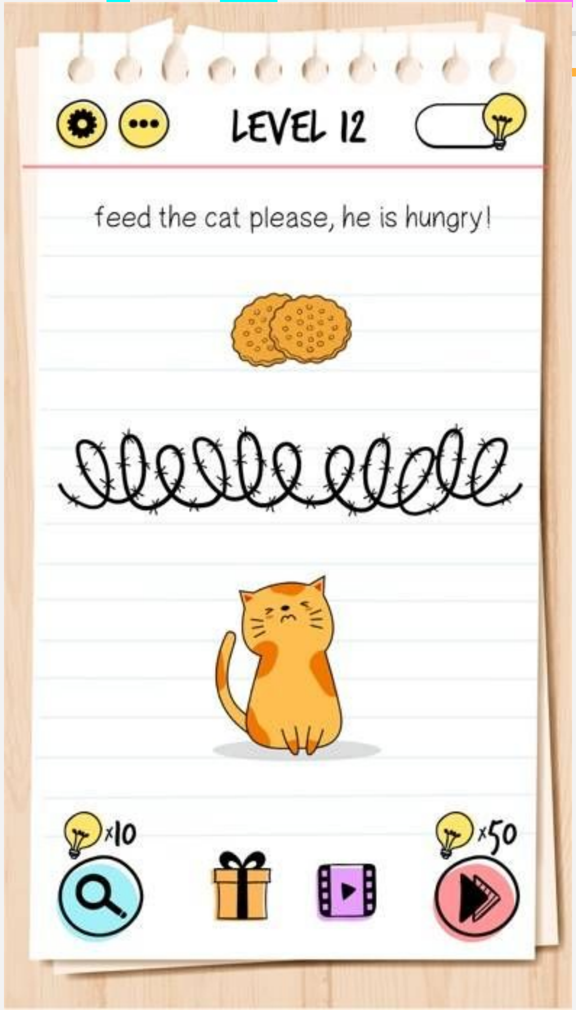
Intuitiveness & Discoverability

Touch interfaces have sometimes been called "**natural**" interfaces

How natural gestures are depends on your unique game, genre, platform, and more.

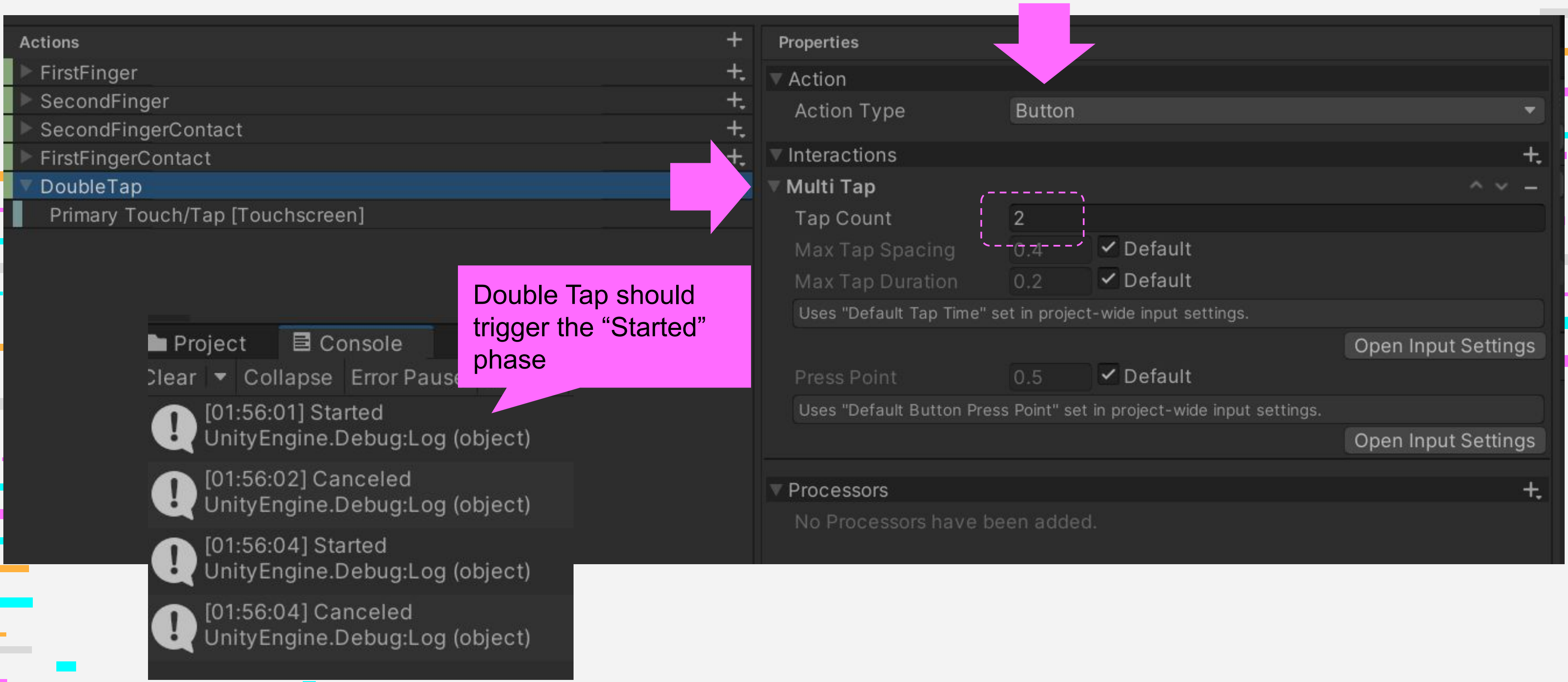
- How would a user every know this gesture is here? How will they learn it? Does it work as part of a tutorial or in context tip?
- If there's no tip or tutorial, how likely is a user to discover this gesture? Will they attempt it on their own?
- How common is a use of this gesture in this type of situation? Is it a convention?


how this truck pass through there?


LEVEL 12

feed the cat please, he is hungry!

DOUBLE TAP



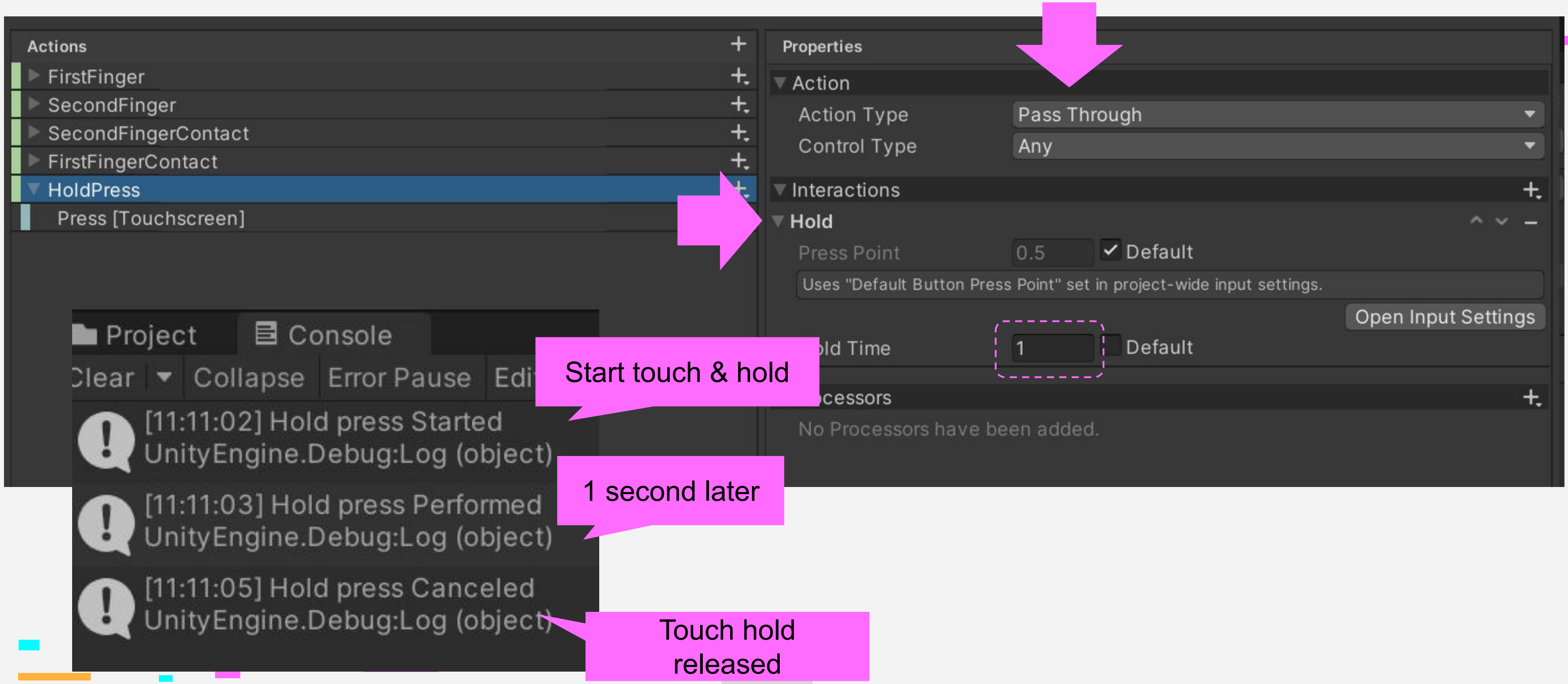
Actions
FirstFinger
SecondFinger
SecondFingerContact
FirstFingerContact
DoubleTap
Primary Touch/Tap [Touchscreen]

Properties
Action
Action Type: Button
Interactions
Multi Tap
Tap Count: 2
Max Tap Spacing: 0.4 ✓ Default
Max Tap Duration: 0.2 ✓ Default
Uses "Default Tap Time" set in project-wide input settings.
Press Point: 0.5 ✓ Default
Uses "Default Button Press Point" set in project-wide input settings.
Processors
No Processors have been added.

```
[01:56:01] Started
UnityEngine.Debug:Log (object)
[01:56:02] Canceled
UnityEngine.Debug:Log (object)
[01:56:04] Started
UnityEngine.Debug:Log (object)
[01:56:04] Canceled
UnityEngine.Debug:Log (object)
```

Double Tap should trigger the “Started” phase

LONG PRESS



Actions

- FirstFinger
- SecondFinger
- SecondFingerContact
- FirstFingerContact
- HoldPress
  - Press [Touchscreen]

Properties

Action

Action Type: Pass Through

Control Type: Any

Interactions

Hold

Press Point: 0.5 ✓ Default

Uses "Default Button Press Point" set in project-wide input settings.

Open Input Settings

Hold Time: 1 Default

Processors

No Processors have been added.

Project | Console

Clear | Collapse | Error Pause | Edit

```
[11:11:02] Hold press Started
UnityEngine.Debug:Log (object)
[11:11:03] Hold press Performed
UnityEngine.Debug:Log (object)
[11:11:05] Hold press Canceled
UnityEngine.Debug:Log (object)
```

Disambiguation

If multiple Controls are bound to an Action, the Input System monitors input from each bound Control to feed the Action.

The Input System must also define which of the bound controls to use for the value of the action.

This Control is the **"driving" Control**; the **Control which is driving the Action**. Unity decides which Control is currently driving the Action in a process called **disambiguation**

Pass-Through

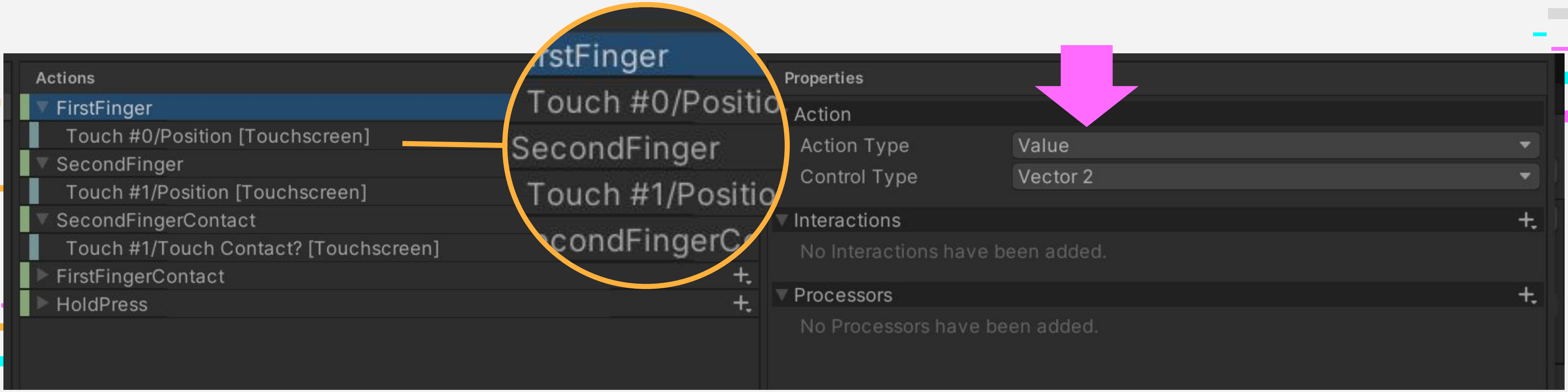
If you don't want your Action to perform disambiguation, you can set your Action type to **Pass-Through**.

Pass-Through Actions skip disambiguation, and changes to any bound Control trigger them. The value of a Pass-Through Action is the value of whichever bound Control changed most recently.

PINCHES

To detect pinches gesture, you need three actions:

1. Monitor First Finger Position
2. Monitor Second FInger Position
3. Detect Second Finger press and release



Actions

- FirstFinger
  - Touch #0/Position [Touchscreen]
- SecondFinger
  - Touch #1/Position [Touchscreen]
- SecondFingerContact
  - Touch #1/Touch Contact? [Touchscreen]
- FirstFingerContact
- HoldPress

Properties

Action

Action Type	Value
Control Type	Vector 2

Interactions

No Interactions have been added.

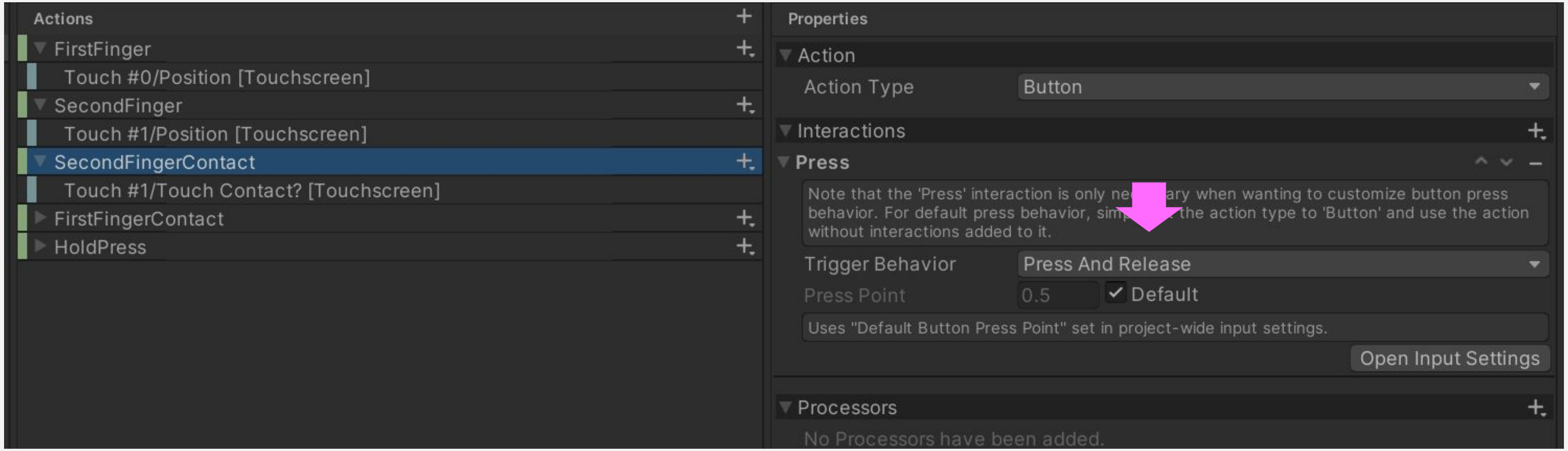
Processors

No Processors have been added.

PINCHES

To detect pinches gesture, you need three actions:

1. Monitor First Finger Position
2. Monitor Second FInger Position
3. Detect Second Finger press and release



Actions
- FirstFinger
  - Touch #0/Position [Touchscreen]
- SecondFinger
  - Touch #1/Position [Touchscreen]
- SecondFingerContact
  - Touch #1/Touch Contact? [Touchscreen]
- FirstFingerContact
- HoldPress

Properties

Action
Action Type: Button

Interactions
Press

Note that the 'Press' interaction is only necessary when wanting to customize button press behavior. For default press behavior, simply set the action type to 'Button' and use the action without interactions added to it.

Trigger Behavior: Press And Release
Press Point: 0.5 ✓ Default

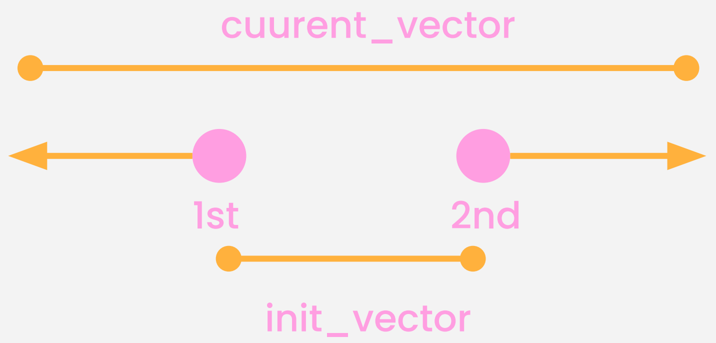
Uses "Default Button Press Point" set in project-wide input settings.

Open Input Settings

Processors
No Processors have been added.

PINCHES ALGORITHM

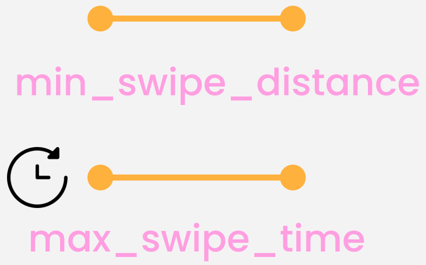
```
If second_finger contact detected then
        Init_vector = second_finger.position - first_finger.position


// IN THE UPDATE STATE
current_vector = second_finger.position - first_finger.position
distance = init_vector.length - current_vector.length

If distance < 0
        initiate gesture pinches_out
else
        initiate gesture pinches_in
```



SWIPE

To detect swipe gesture, you need three actions:

1. Monitor Finger Position
2. Detect Finger press and release



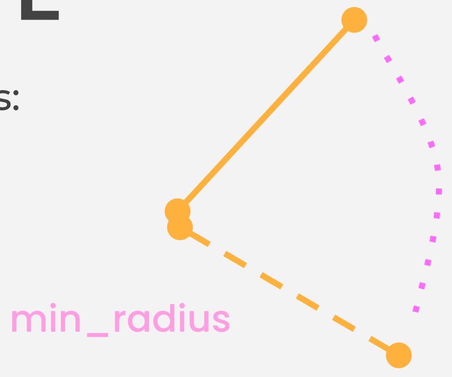
Swipe Algorithm

```
If finger press then
        vector_touch_start = finger.position
        touch_start = current_time
else if finger release then
        vector_touch_end = finger.position
        touch_end = current_time

        if ABS(vector_touch_start.x - vector_touch_end.x) > min_swipe_distance then
                If touch_start - touch_end  < max_swipe_time then
                        trigger swap (valid)
```

ROTATE

To detect rotate gesture, you need three actions:

1. Detect Second Finger press and release
2. Monitor First Finger Position
3. Monitor Second Finger Position



Rotate Algorithm

```
If second_finger press then
        start_vector = second_finger.position - first_finger.position

// UPDATE
        current_vector = second_finger.position - first_finger.position
        angle_offset = findAngle( start_vector,  current_vector)

        If current_vector.magnitude > min_radius then
                Initiate gesture rotate (valid)
```

WEEK 5 ASYNCHRONOUS - VIDEO

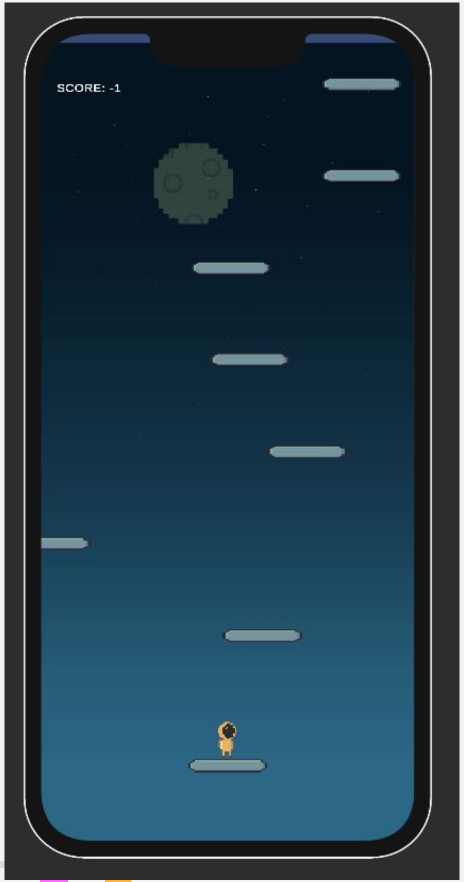
Learn how to work with:

- Pinches Gesture
- Swipe Gesture

WEEK 5 - CLASS TUTORIAL

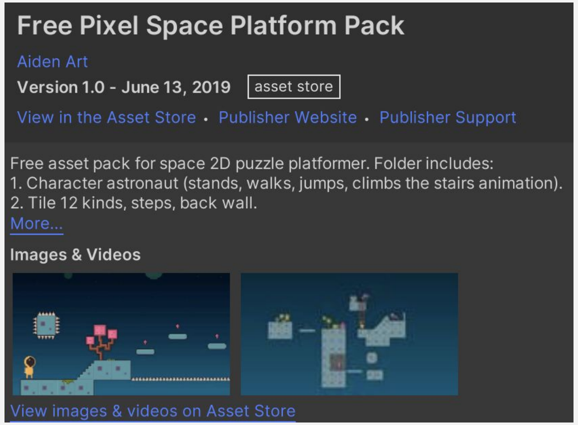
Learn how to work with:

- Double Tap to toggle zoom
- Swipe to jump



GAME SETUP

- This section provide how to setup a new game project
- Create a new 2D Core Project
- Name it as “DoodleJump”
- Import Cinemachine
- Import Input System
- Import Free Pixel Space Platform Pack from Unity Store



Free Pixel Space Platform Pack

Aiden Art

Version 1.0 - June 13, 2019 asset store

View in the Asset Store • Publisher Website • Publisher Support

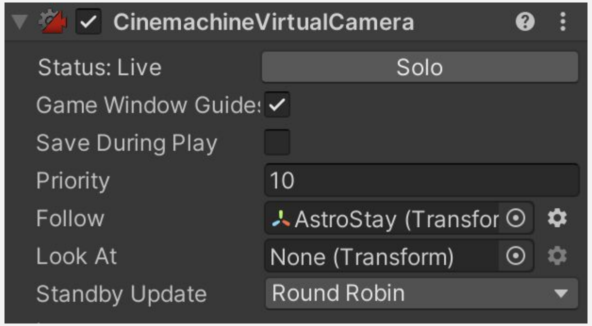
Free asset pack for space 2D puzzle platformer. Folder includes:
1. Character astronaut (stands, walks, jumps, climbs the stairs animation).
2. Tile 12 kinds, steps, back wall.
More...

Images & Videos

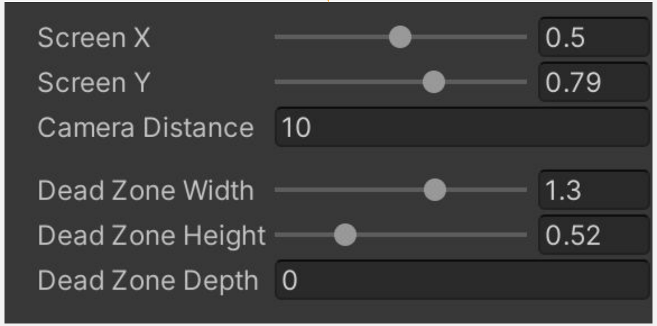
View images & videos on Asset Store

SETUP CINEMACHINE

- Cinemachine 2D camera is useful to track the game object Orthographically in 2D plane
- Click GameObject menu > Cinemachine > 2D Camera
- Drag and drop “AstroStray” gameobject to Follow property in the inspector. This will make cinemachine camera follow the game object
- Find and adjust following properties in the inspector


CinemachineVirtualCamera

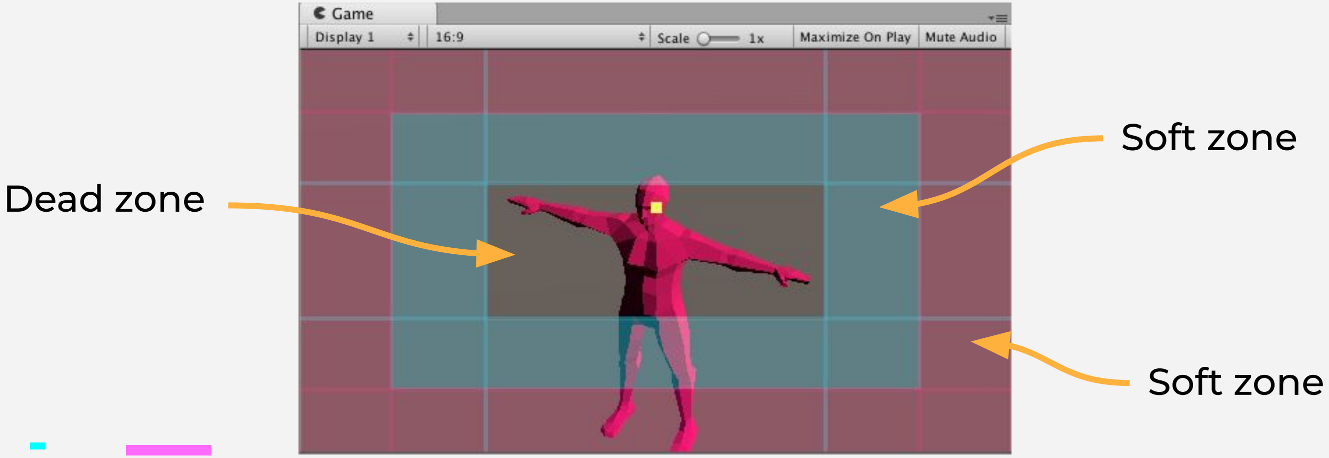
Status: Live	Solo
Game Window Guide:	✓
Save During Play	
Priority	10
Follow	AstroStay (Transfor
Look At	None (Transform)
Standby Update	Round Robin



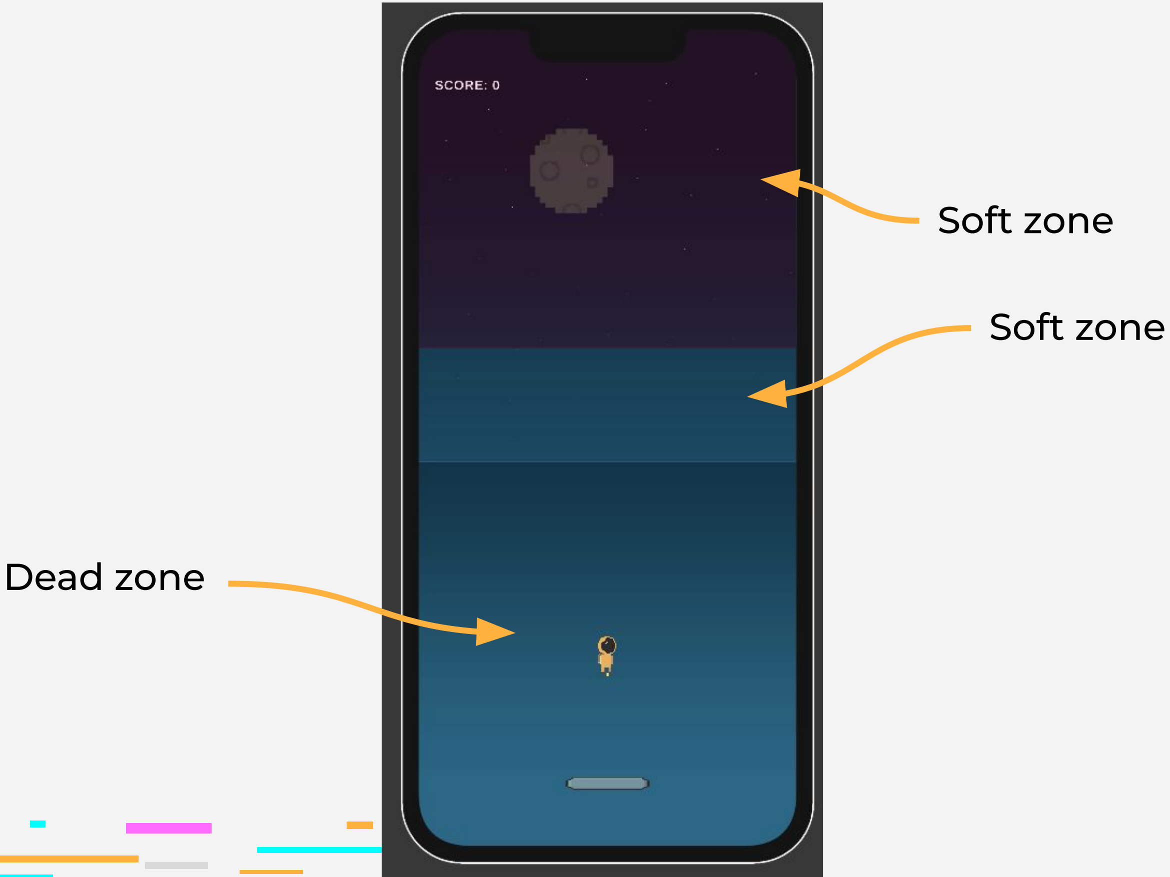
Screen X	0.5
Screen Y	0.79
Camera Distance	10
Dead Zone Width	1.3
Dead Zone Height	0.52
Dead Zone Depth	0

DEAD ZONE

- Dead zone is the area of the frame that Cinemachine keeps the target in
- The clear area indicates the dead zone. The blue-tinted area indicates the soft zone. The position of the soft and dead zones indicates the screen position. The red-tinted area indicates the no pass area, which the target never enters. The yellow square indicates the target.

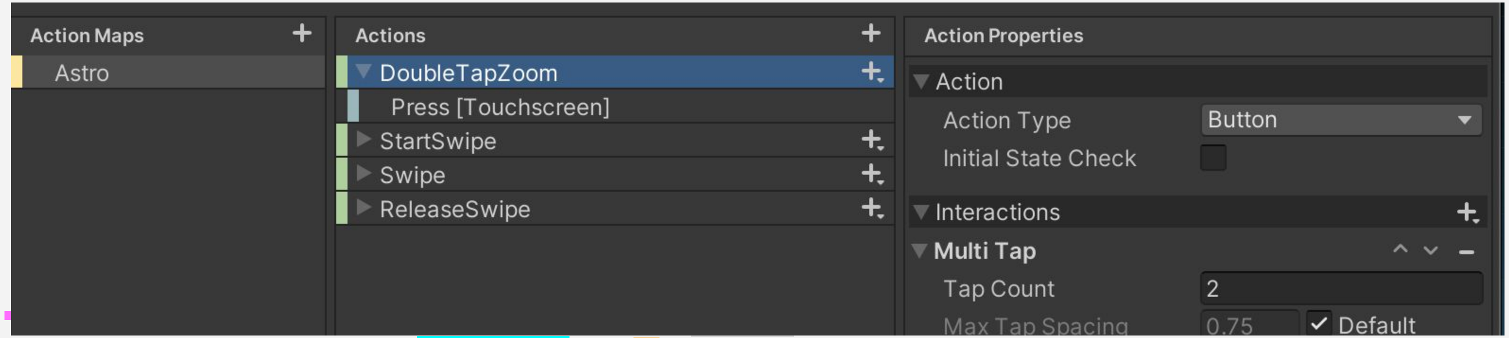


DEAD ZONE



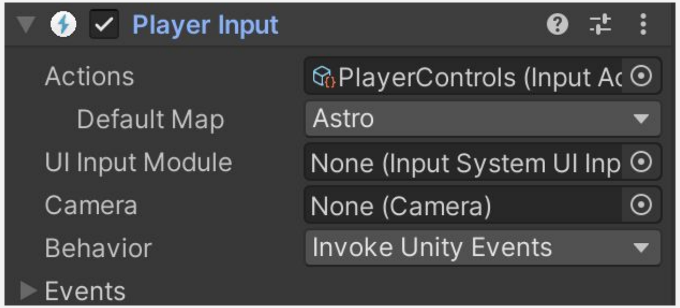
PLAYER ACTIONS - DOUBLE TAP ZOOM

- Create a new folder “Scripts” in “Asset” folder
- Right click and choose “Input Actions”, name it as “PlayerControls”
- Implement double tap zoom toggle
- Action Maps -> “Astro”
- Actions -> “DoubleTapZoom”. Action Type to “Button”. Add interaction “Multi Tap”
- Binding to “Press Touchscreen”
- Press save asset


Action Maps
Astro
Actions
DoubleTapZoom
Press [Touchscreen]
StartSwipe
Swipe
ReleaseSwipe
Action Properties
Action
Action Type: Button
Initial State Check
Interactions
Multi Tap
Tap Count: 2
Max Tap Spacing: 0.75 ✓ Default

ASTRO GAME OBJECT

- Drag and Drop "AstroStray" from Free Pixel asset
- Scale it to 0.1, 0.1, 1
- Reset to 0,0,0
- Add PlayerInput component
- Drag and drop "PlayerControls" to "Action" in PlayerInput component
- Change Behavior to "Invoke Unity Events"



Player Input

Actions	PlayerControls (Input Ac
Default Map	Astro
UI Input Module	None (Input System UI Inp
Camera	None (Camera)
Behavior	Invoke Unity Events

Events

ASTRO HANDLER SCRIPT

- Next, create logic code for handling the double tap zoom action
- Create a new C# script, name it as “AstroHandler”
- Create three data members/fields:

```
[SerializeField]
private CinemachineVirtualCamera vCam;
private float zoomSpeed = 10.0f;
private bool zoomIn = false;
```

Variable to get virtual camera object

Indicate zoom speed and flag to
handle toggle between zoom-in/false

ASTRO HANDLER SCRIPT

```
public void OnDoubleTapZoom(InputAction.CallbackContext ctx)
{
   // Debug.Log(ctx.phase);
    if(ctx.phase == InputActionPhase.Performed)
    {
        if(zoomIn)
        {
            zoomIn = false;
        } else
        {
            zoomIn = true;
        }
    }
}
```

OnDoubleTapZoom is method callback to handle input action

InputAction phase performed occurred if double tap successfully catched

Toggle zoomIn variable

ASTRO HANDLER SCRIPT

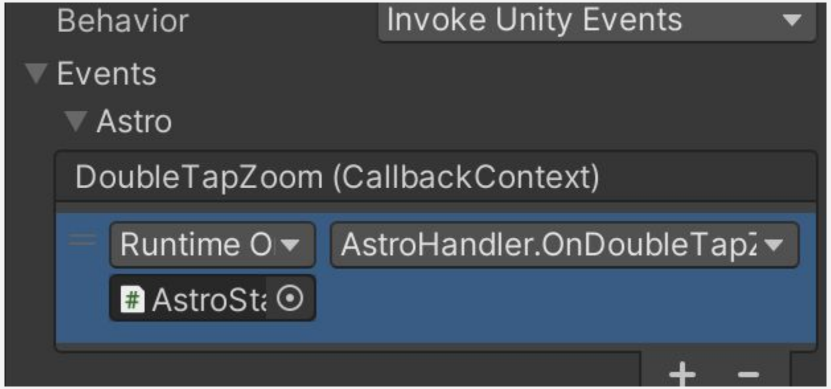
```
// Update is called once per frame
void Update()
{
   if(zoomIn && vCam.GetComponent<CinemachineVirtualCamera>().m_Lens.OrthographicSize >= 1.0f) {
      vCam.GetComponent<CinemachineVirtualCamera>().m_Lens.OrthographicSize -= zoomSpeed * Time.deltaTime;
   } else if(!zoomIn && vCam.GetComponent<CinemachineVirtualCamera>().m_Lens.OrthographicSize <= 5.0f) {
      vCam.GetComponent<CinemachineVirtualCamera>().m_Lens.OrthographicSize += zoomSpeed * Time.deltaTime;
   }
}
```

In update method, adjust the virtual camera lens ortohgraphic size. This property handle zoom in/out.

Change lens property according to zoom speed

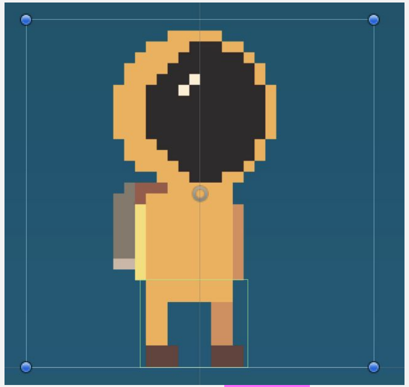
ATTACH THE FUNCTION

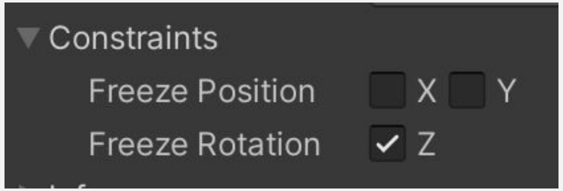
- Drag and Drop the Astroy Handler script into AstroStray game object
- Drag and Drop the AstroStray game object into DoubleTapZoom callback in property inspector of “PlayerInput”
- Choose DoubleTapZoom
- Run the game and double tap to toggle zoom in and out


Behavior: Invoke Unity Events
Events
Astro
DoubleTapZoom (CallbackContext)
Runtime O | AstroHandler.OnDoubleTapZ
AstroSt

IMPLEMENT SWIPE MECHANIC

- Next, we will implement swipe mechanic to make character jump between platforms
- Add BoxCollider2D to AstroStray game object
- Adjust the collider like following image
- Add RigidBody2D to AstroStray game object
- Tick "Freeze Rotation" in z-axis constraint to prevent game object rotating in Z axis

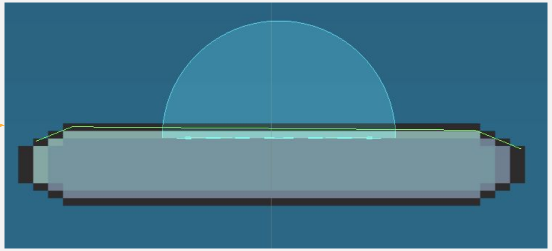


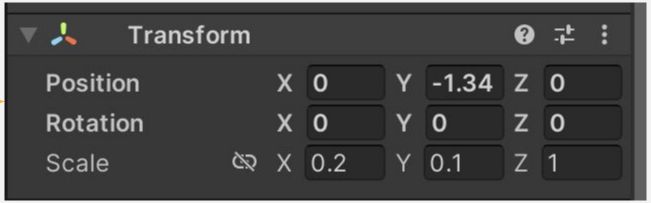


Constraints
Freeze Position [ ] X [ ] Y
Freeze Rotation [✓] Z

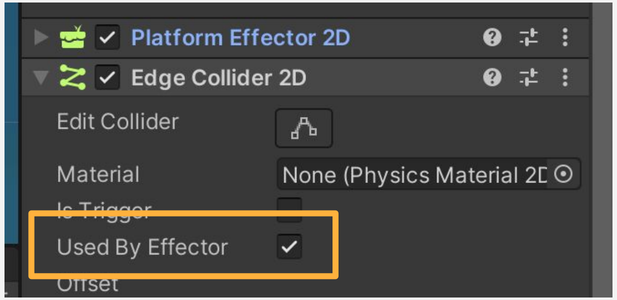
PLATFORMS

- Drag and Drop platforms asset (Platform_01) into hierarchy
- Adjust its transformation like following image
- Add PlatformEffector2D component
- Add EdgeCollider2D component
- Make sure "Used by Effector" is ticked
- Edit Collider like following image



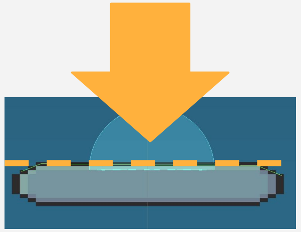

Transform

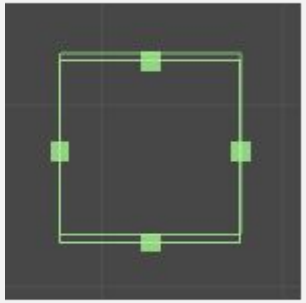
Position	X	0	Y	-1.34	Z	0
Rotation	X	0	Y	0	Z	0
Scale	X	0.2	Y	0.1	Z	1

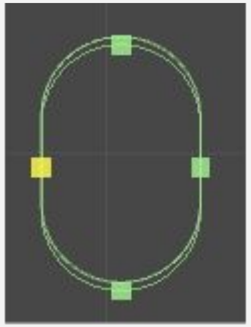

Platform Effector 2D ✓
Edge Collider 2D ✓
Edit Collider
Material: None (Physics Material 2D
Is Trigger: ☐
Used By Effector: ✓
Offset

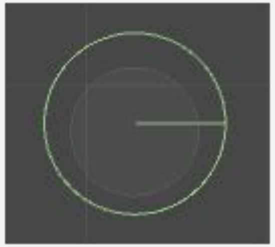
EDGE COLLIDER 2D

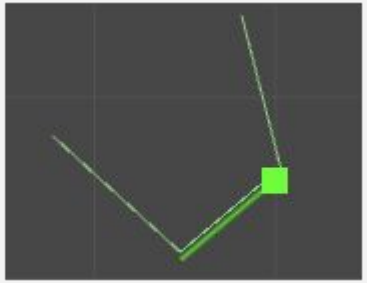
- The Collider's shape is a freeform edge made of line segments that you can adjust to fit the shape of a Sprite or any other shape.
- We use EdgeCollider 2D to only allow collision from above game object (not below)

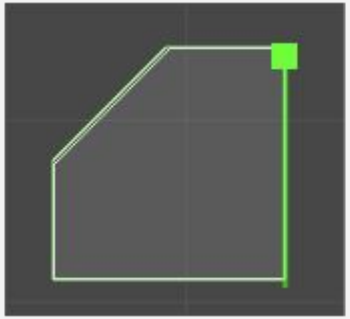



Box Collider 2D


Capsule Collider 2D


Circle Collider 2D


Edge Collider 2D

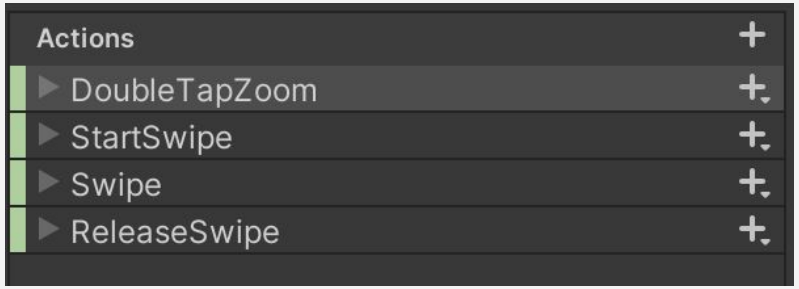

Polygon Collider 2D

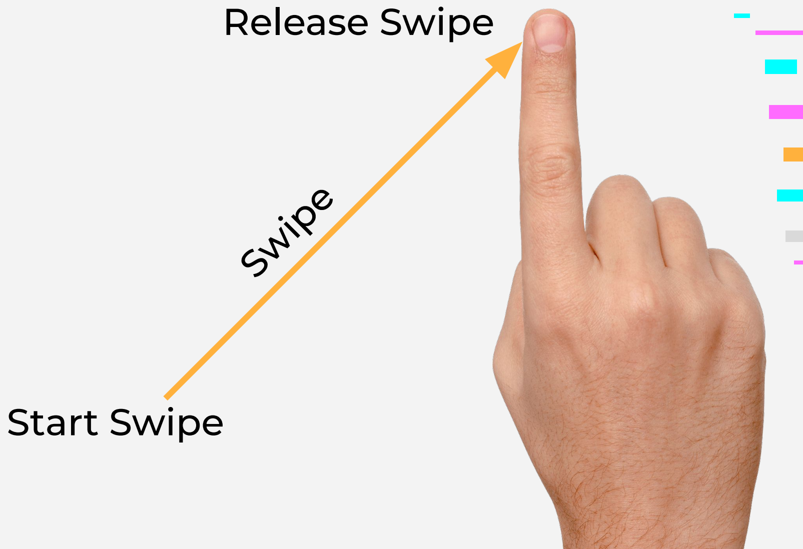
PREFAB

- Create “Prefab” folder in Asset
- Drag and drop the platform game object into this folder to make this object to prefab

SWIPE ACTION

- We define three actions:
  - StartSwipe -> triggered to capture the start position of swipe
  - Swipe -> triggered when player move finger across screen surface (to capture screen coordinate)
  - ReleaseSwipe -> triggered when player release finger


Actions
DoubleTapZoom
StartSwipe
Swipe
ReleaseSwipe



SWIPE ACTION

- StartSwipe:
    - Action type: Value
    - Control Type: Vector2
    - Binding: Start Position - Primary Touch - Touchscreen
- Swipe:
    - Action type: Value
    - Control Type: Vector2
    - Binding: Position - Primary Touch - Touchscreen
- ReleaseSwipe
    - Action type: Button
    - Control Type: Vector2
    - Binding: Press (singe touch) - Touchscreen

Dont forget to save asset

ASTRO HANDLE SCRIPT

- Add following data members/fields:

```
private Vector3 startPos;
private Vector3 currentPos;
```

startPos variable is used to record first touch contact position

currentPos is used to record current finger position on screen

- Add following method:

```
private void OnBecameInvisible()
{
        Scene scene = SceneManager.GetActiveScene();
        SceneManager.LoadScene(scene.name);
}
```

The OnBecameVisible is triggered when player move off screen

```
public void OnStartSwipe(InputAction.CallbackContext ctx)
{        startPos = ctx.ReadValue<Vector2>();   }
```

The OnStartSwipe is used to record startPos

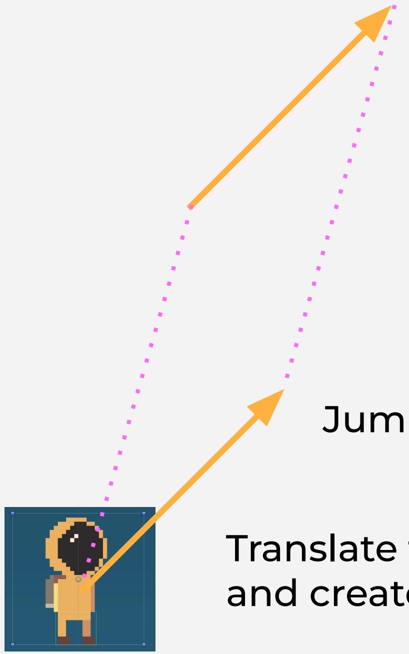
ASTRO HANDLE SCRIPT

- Add following data members/fields:

```
public void OnSwipe(InputAction.CallbackContext ctx)
{
     currentPos = ctx.ReadValue<Vector2>();
}

public void OnReleaseSwipe(InputAction.CallbackContext ctx)
{
        if(ctx.phase == InputActionPhase.Canceled)
        {
            var distance = startPos - transform.position;
            var jumpVector = currentPos - distance;
            GetComponent<Rigidbody2D>().AddForce(jumpVector/10.0f * 5f);
         }
}
```

Callback method when finger move across screen surface. It captures finger position

ASTRO HANDLE SCRIPT



Player swipe here

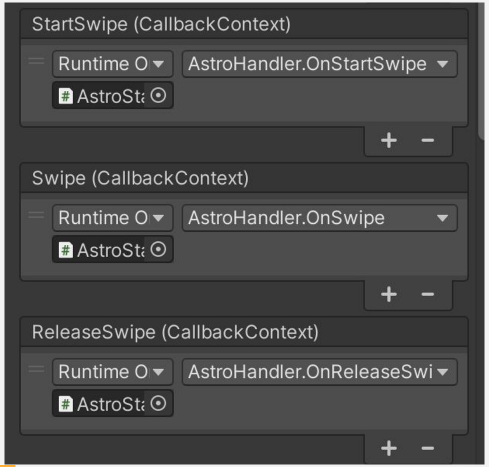
Jump Vector

Translate to gameobject origin
and create jump vector

Apply this jump vector to add
force for game object rigidbody

ATTACH THE EVENT

- Attach the StartSwipe, Swipe, and ReleaseSwipe into AstroStray inspector
- Run the game, and try to swipe to make character jumps
- Notice that faster you swipe, the higher character would jump



StartSwipe (CallbackContext)

Runtime O | AstroHandler.OnStartSwipe
AstroSt

Swipe (CallbackContext)

Runtime O | AstroHandler.OnSwipe
AstroSt

ReleaseSwipe (CallbackContext)

Runtime O | AstroHandler.OnReleaseSwi
AstroSt

INSTANTIATE PLATFORMS

- Create empty game object, name it as “Game Manager”
- Create C# script, name it as “GameManager”
- Attach this script into game object above
- Add following script into class data member/fields:

```
[SerializeField]
private GameObject platformRefPoint;

[SerializeField]
private GameObject platformPrefab;
```

Attach platform game object from hierarchy

Attach platform prefab

INSTANTIATE PLATFORMS

```
void Start() {
     // Get platform width in screen space coordinates
      var refPoint = Camera.main.WorldToScreenPoint(platformRefPoint.transform.position);
      var min = platformRefPoint.GetComponent<SpriteRenderer>().bounds.min;
      var max = platformRefPoint.GetComponent<SpriteRenderer>().bounds.max;
      var lebar = max - min;
```

INSTANTIATE PLATFORMS

```
void Start() {
     . . .

     for (var i=1; i <= 100; i++)
     {
        var x = Random.Range(- Camera.main.WorldToScreenPoint(lebar).x, Screen.width +
               Camera.main.WorldToScreenPoint(lebar).x);
        var randomPos = Camera.main.ScreenToWorldPoint(new Vector3(x,
               refPoint.y+(i*1500.0f)),0f);

        Instantiate(platformPrefab, randomPos, Quaternion.identity);
     }
}
```

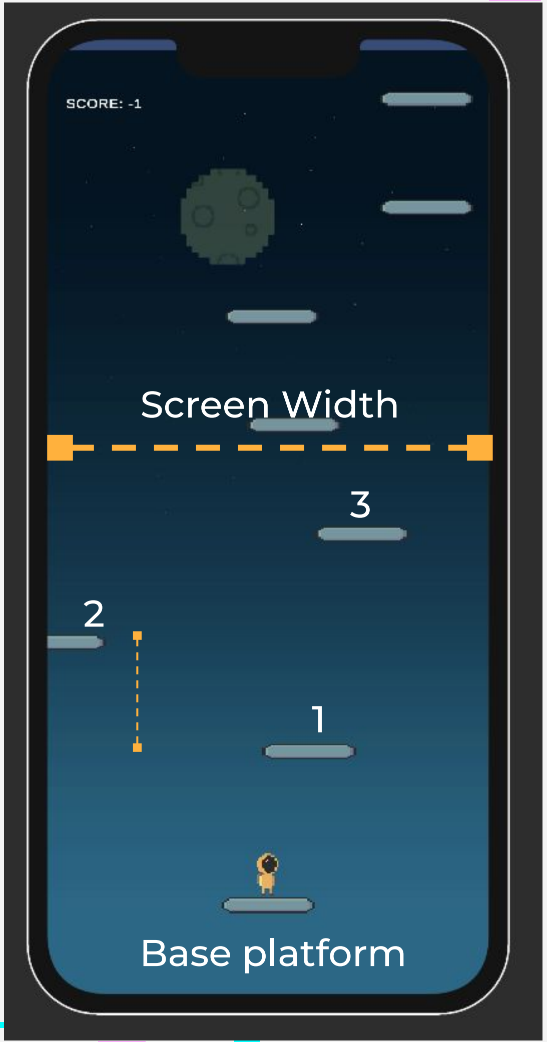
Create 100 platforms

Randomize horizontal position along the screen width

Find y position by using iterator i. Use first platform as references.

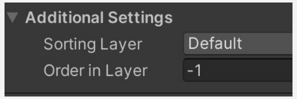
Instantiate the platform

Offset the screen width with platform “lebar” to precisely put platform within screen.



EXERCISE

- Place SpaceBackground to MainCamera, make sure it negative value in "Order in Layer"



Additional Settings

Sorting Layer	Default
Order in Layer	-1

- Score UI:
  - Add UI Screen to store "SCORE"
  - Score = AstroStray y position (vertical position)
  - Show the score to UI
- Implement swipe timing logic:
  - Reduce the jumpforce strength if swipe gesture is slow
  - To solve this you need to record time between start swiping until finger is released
  - Based on the swipe duration, you can determine the jumpforce strength

SUBMISSION

- Submit your works to ULS
- What to submit:
  - A screenshot of your Unity UI that shows hierarchy and game scene
  - A screenshot of game simulator scene to show scores changes
  - All the C# scripts in the Asset > Scripts folder
  - Compress all above into single zip files
- Deadline: today

THANKS!

Do you have any questions?
youremail@freepik.com
+91 620 421 838
yourcompany.com

slidesgo