2.Basic programming with Unity

- This course aims at learning the basics of programming with Unity.
- Unity allows beginners to create own games easily.
- Have fun for programming!

LET'S START

Our Goals

What you can after this course...

1. Understand basic of programming on C#.
2. Understand the role of variables, functions and classes and how they can be used to create efficient code.
3. Able to design simple classes.
4. Able to edit simple scripts and test the impact of changes on the player experience.
5. Able to write your own code using C#.

Class Plan

1. **Introduction**
2. Hello world in Unity
3. Basic programming
4. Object oriented programming
5. Interface programming
6. Programming design
7. Programming design 2
8. Improvement and testing 1
9. Improvement and testing 2
10. Summary

1
Introduction

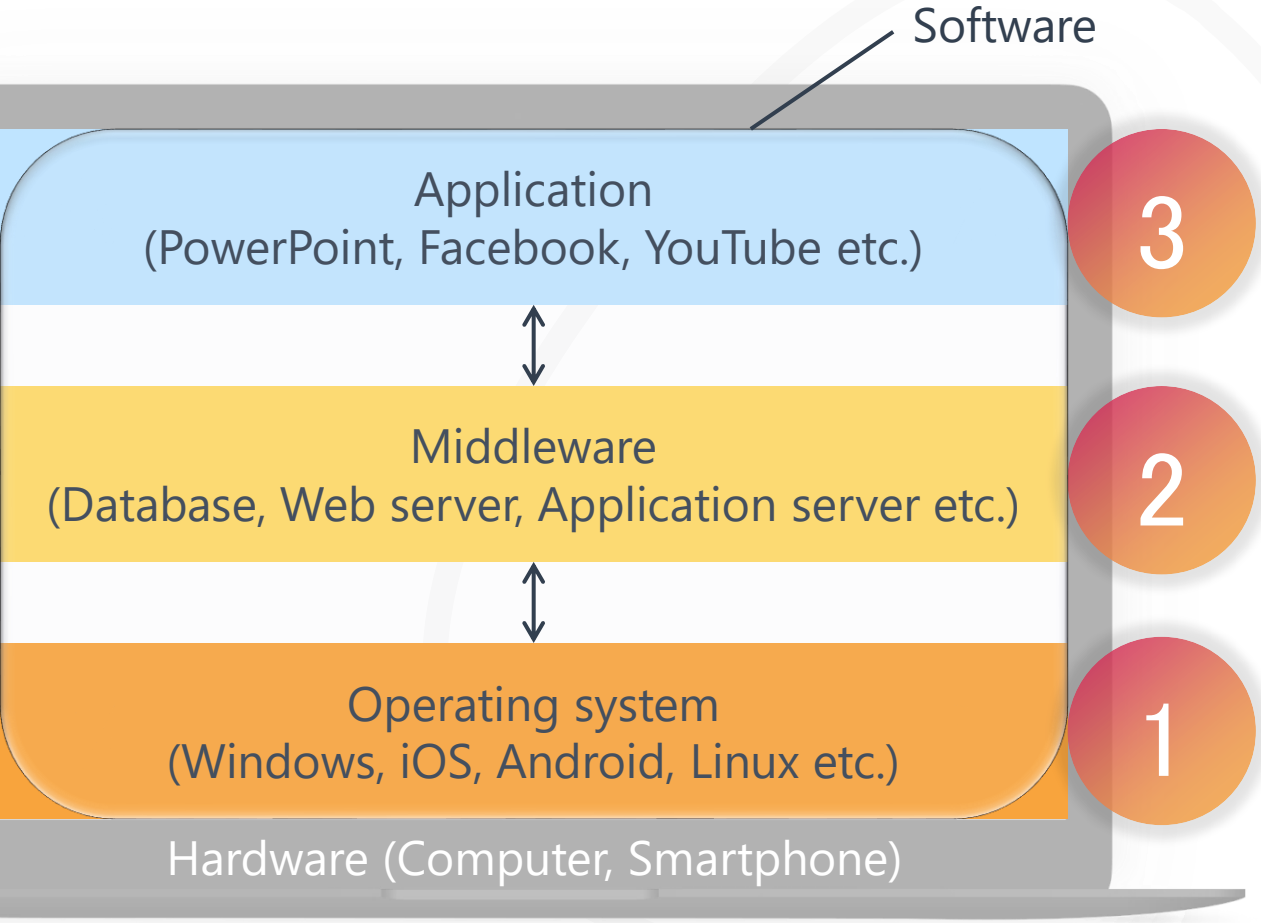
1. Introduction

Class summary

1. Explain the basic knowledge of programming.
2. Explain the programming languages.
3. Explain the process of game and software development.
4. Set up the software used in this course.
5. Explain the basics of Unity.
6. Explain the basic syntax on programming.

What is Programming?

Writing instructions to drive software



3 layer of the Software

Application

Generally, users only can use this layer's software directly to fill ther porpose. In this cource, we will develop it by programming.

3

Middleware

Middleware supports to handle data exchange between Application and Operating System.

2

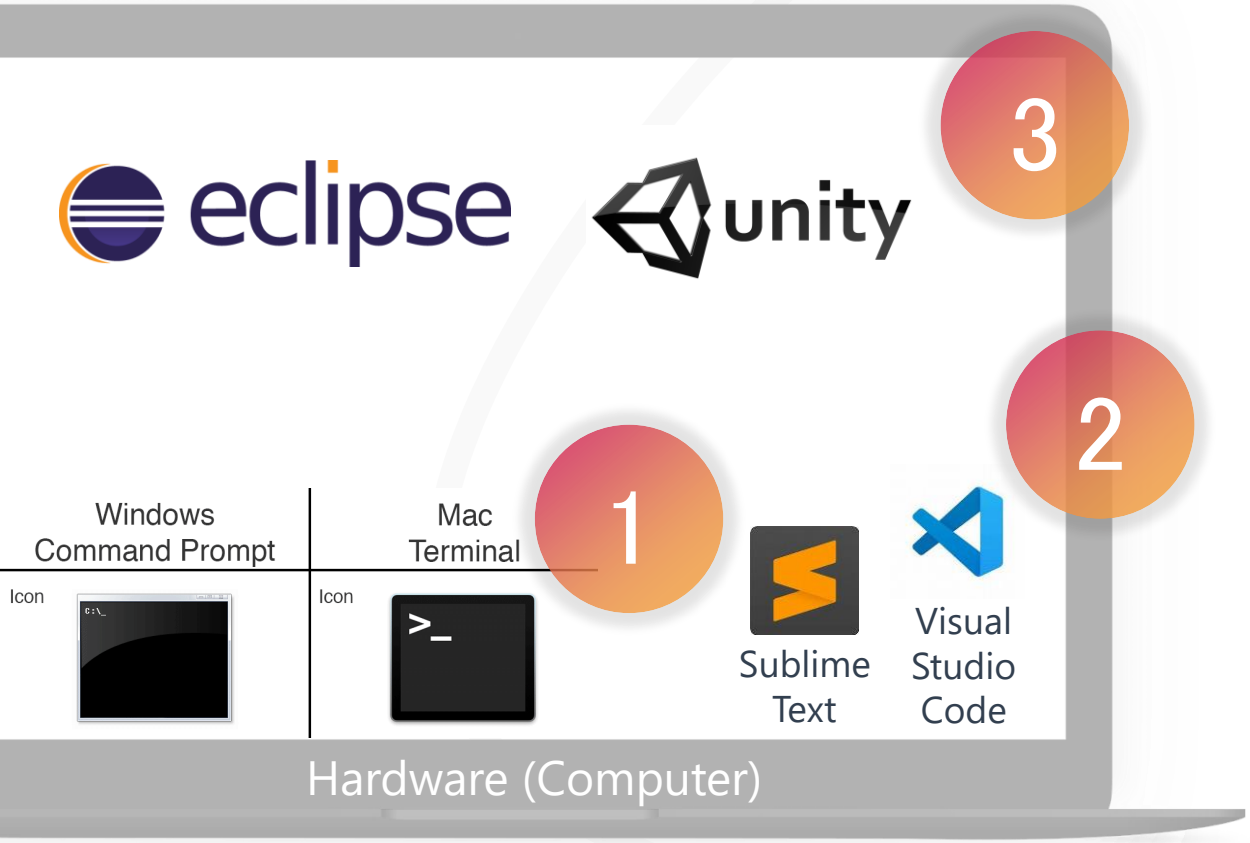
Operating System

OS aims to abstract hardware, manage resources, and improve the efficiency of computer usage.

1

What is Programming?

Writing instructions to drive software



3 type of tools for programming

Integrated Development Environment

IDE is consist of a set of various tools like Programming Editor, Compiler and debuger etc.

3

Programming Editor

Tools to write program efficiently. It supports GUI: Graphical User Interface, input completion and customize etc.

2

Command Prompt / Terminal

Text (command) based interface to manipulate files on computer. It is also called command line interface. Most primitive control method.

1

What is Programming?

Writing instructions to drive software

Program

- Program needs compile to translate machine language.
- The pre-translation provides faster execution.
- It is used to embedded system like OS, car mozule, smartphone, imaging solution, AI and big data etc.
- Languages: C++, C#, Go, Java, Swift etc.

```
#include <stdio.h>

int main(void) {
  printf("Hello World¥n");
  return 0;
}
```

compile

```
0011 0101 0111 1111 0110
0001 1000 0101 1010 1100
1110 1011 0111 0001 0011
1101 0111 1011 1110 0010
```

Source code

Interpriter

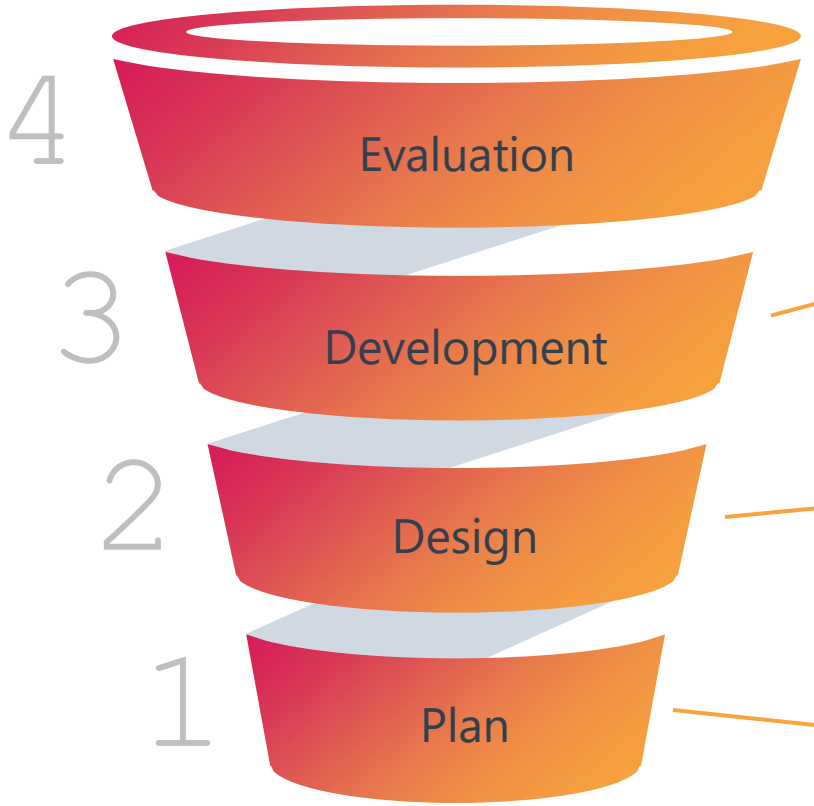
```
<!doctype html>
<html>
 <head>
   <meta charset="utf-8">
   <title>sample</title>
 </head>
<body>
```

Script

- Program needs interpriter to translate machine language.
- The sequential-translation provides faster testing because users do not need waiting compile.
- It is used to web service etc.
- Languages: Javascript, PHP, Phython, Ruby etc.

Development Process

Waterfall and spiral model



- Waterfall: Conduct 1 -> 4 at once.
- Spiral: Conduct 1 or 2 -> 4 repeatedly.

Evaluation / Assessment

Test your code according to documents. How do you find bugs effectively?

Development / Coding

Type code according to design. How do you create beautiful and efficient code?

Design

Express what function do you need to develop it. How do you realize it?

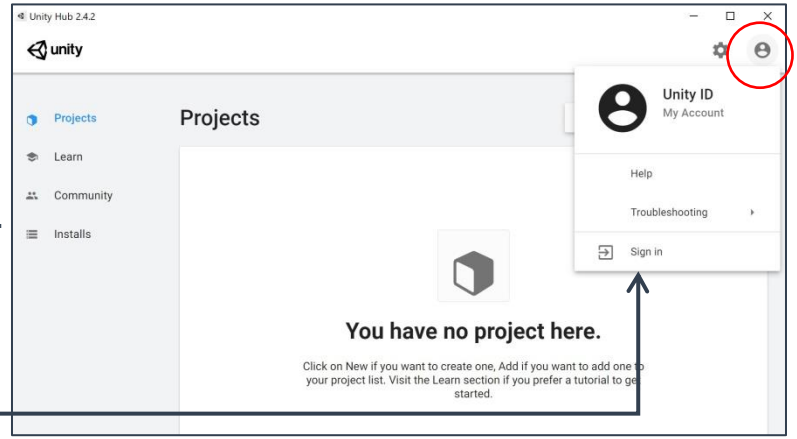
Plan

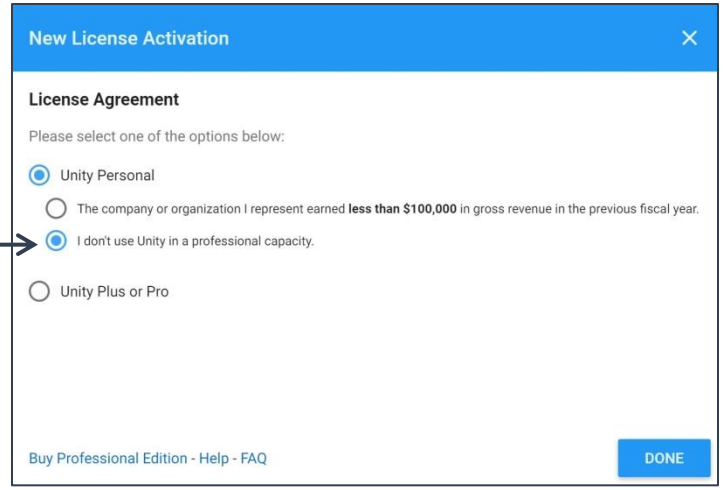
Determine what porpose do you want to meet. How do you satisfy users?

Set up Unity

To create your game

1. Download and Install Unity Hub: Unity's version management system.
    a. Access: https://unity3d.com/jp/get-unity/download
    b. Agree to Unity Terms of Service.
    c. (Windows only) Select install folder. No problem with the default.
2. Activate with free Unity license.
    a. Execute Unity hub.
    b. Create Unity account or sign in via top-right profile icon.
    c. Select ACTIVATE NEW LICENSE in Preferences -> License management.
    d. Select "Unity personal" and "I don't use Unity in a professional capacity."
    e. Select DONE .



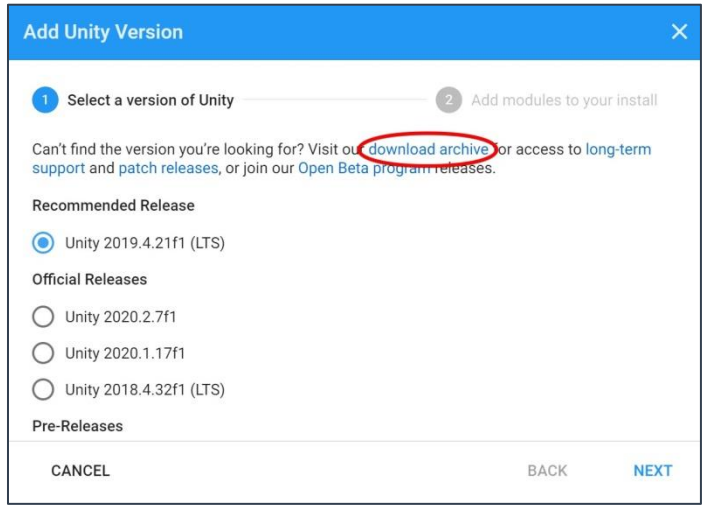


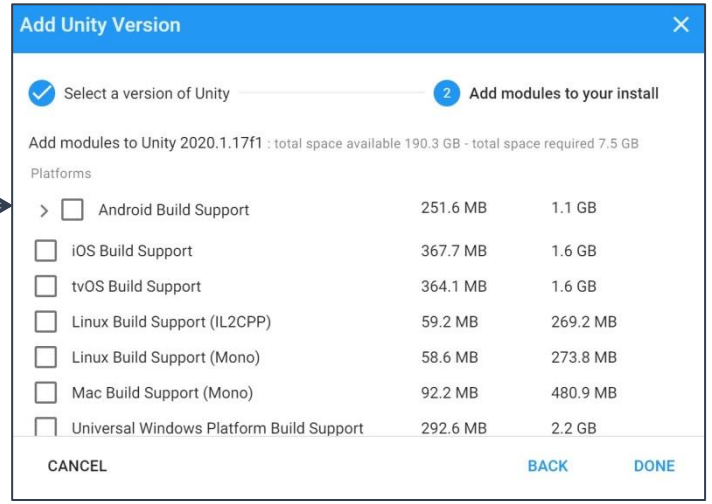
Set up Unity

To create your game

3. Install Unity (and Visual Studio 2019 community).
   a. Back to top from Preferences and select “Installs” -> ADD .
   b. Select “download archive”.
   c. Select Unity Hub on “Unity 2020.2.0”
   d. ⚠ (Mac Only) Check “Visual studio for mac”. Documentation and Language packs are optional.
   e. Select DONE .
   f. Check terms and select DONE .
   g. Wait finishing the download.

This mark means it depends on your environment. I follow up with it on this class.

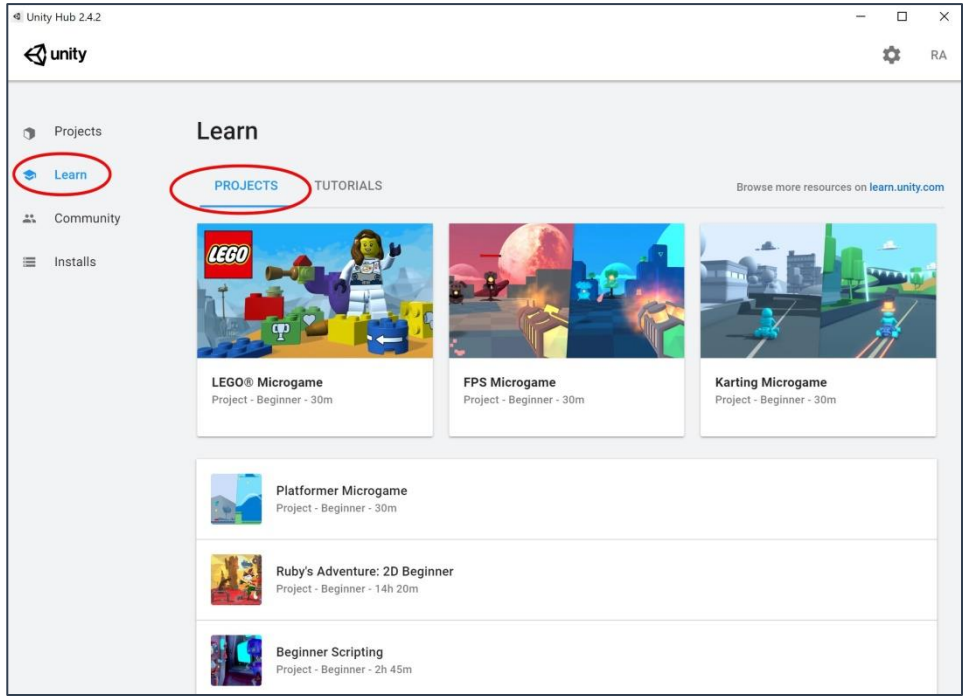


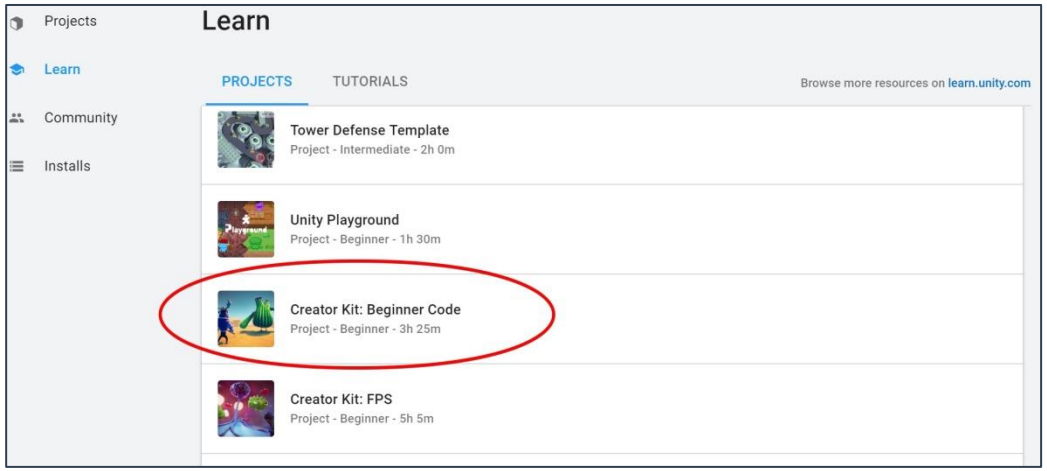


Set up Unity

To create your game

5. After finishing Install, set up tutorial project.
    a. Select “Learn” -> “PROJECTS” tab.
    b. Select “Creator Kit: Beginner Coding”.
    c. ⚠ Select “DOWNLOAD PROJECT”.
    d. ⚠ Wait finishing the download.
    e. ⚠ Select “OPEN PROJECT”.
    Wait a few minutes.
    f. This instruction is written in official page:
    https://learn.unity.com/project/creator-kit-beginner-code
6. Name and save the project.
    - File -> Save as -> save it as “BeginnerCode”.
    - When you save the project, please use ctrl+s (Wndows) or cmd+s (Mac). Useful.
    - If you have problem to save, it may be error from Unity. Please check here and try it.
    - It is recommended to create new folder like “unity/projects” to manage it.

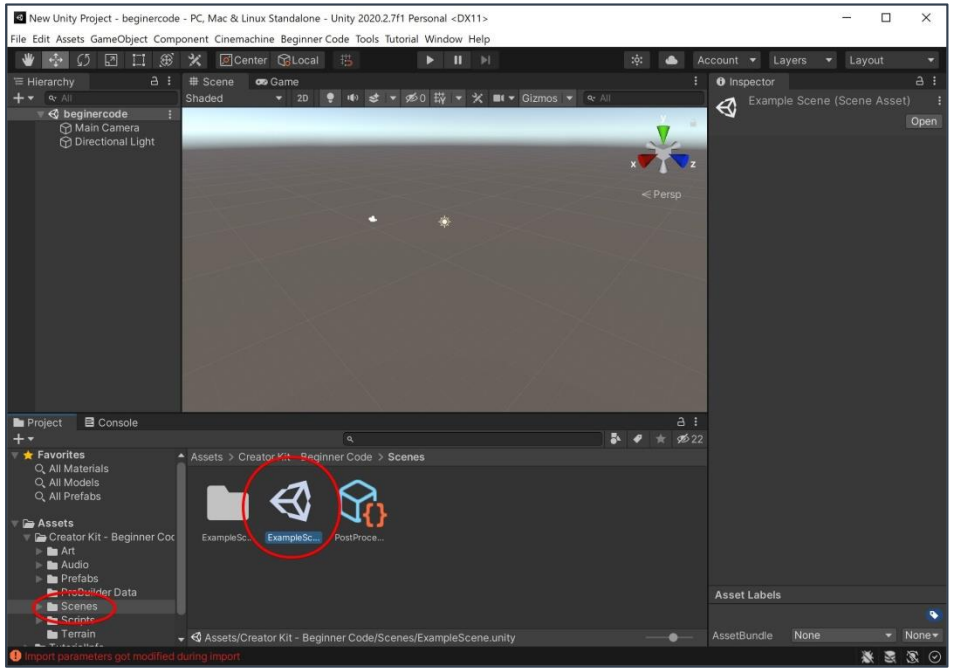


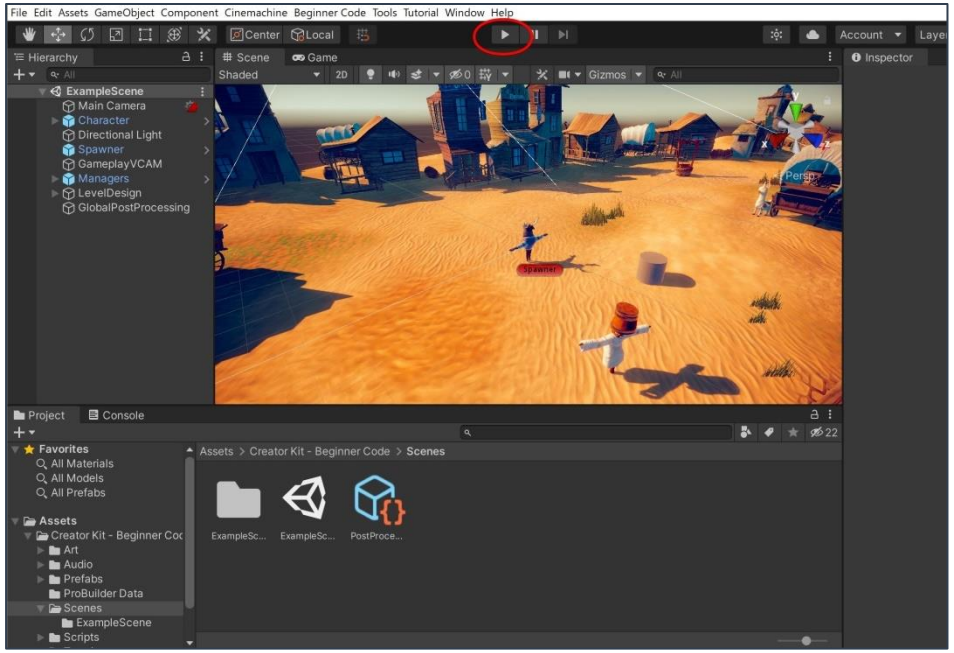


Set up Unity

To create your game

7. Play sample game.
    a. Select below folder.
    “Assets/Creator Kit – Beginner Code/Scenes”
    b. Double click “ExampleScene”.
    c. Select ▶ “Play” button in top-middle.
    d. Start the game.
        - Move character by click.
        - Get Potions on the ground.
        - Open inventory by “i” key or bag icon.
        - Explore and Beat enemies!
        - Use portions by double click on inventory.
        - Find new weapon.
        - End game with Play button again.



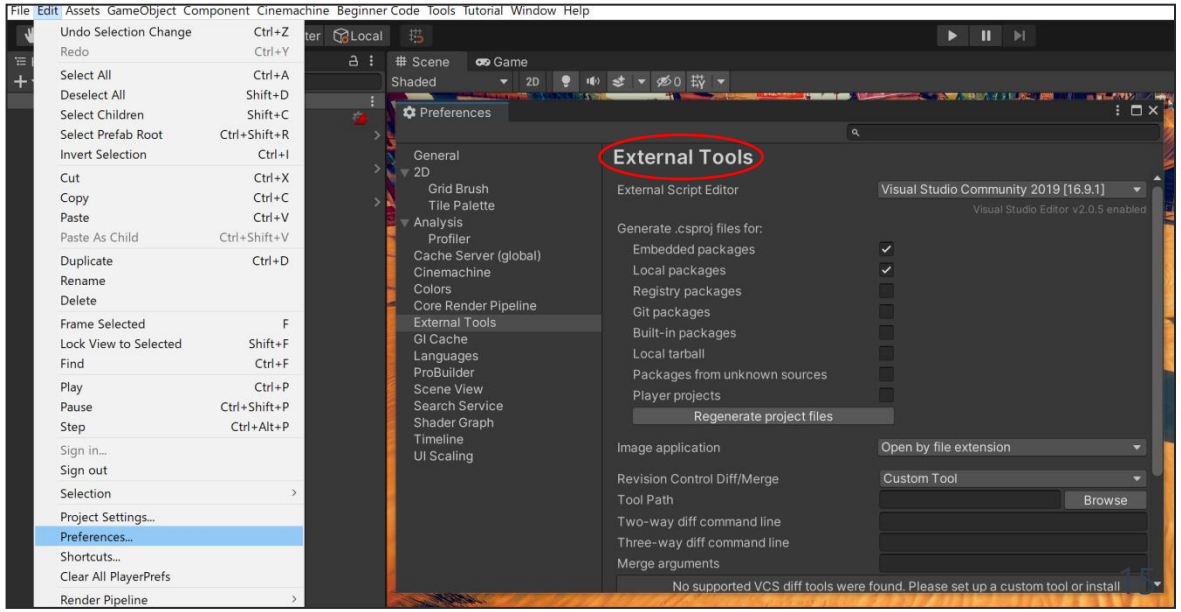


Set up Visual Studio

To create your game

1. (Windows only) Install Visual Studio community.
    a. Exit Unity.
    b. Download and execute installer.
    https://visualstudio.microsoft.com/ja/downloads/
    c. Select Game Development option.
    d. Deselect Unity hub checkbox and install.
    e. Skip or sign in with Microsoft account.
    f. Proceed as default settings.

2. (Windows only) Set up Visual Studio in Unity.
    a. Execute Unity Hub.
    b. Open “beginnercode” project.
    c. Select “Edit” -> “Preferences…” -> “External tools”.
    d. Select “Visual Studio Community 2019” on “External Script Editor”.



What is Unity?

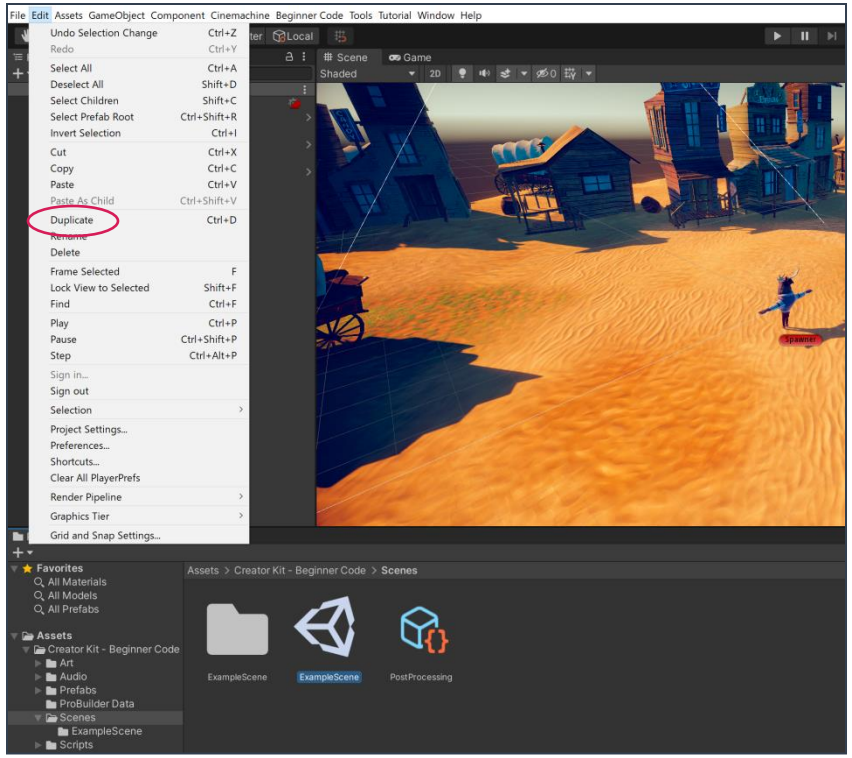
To create your game

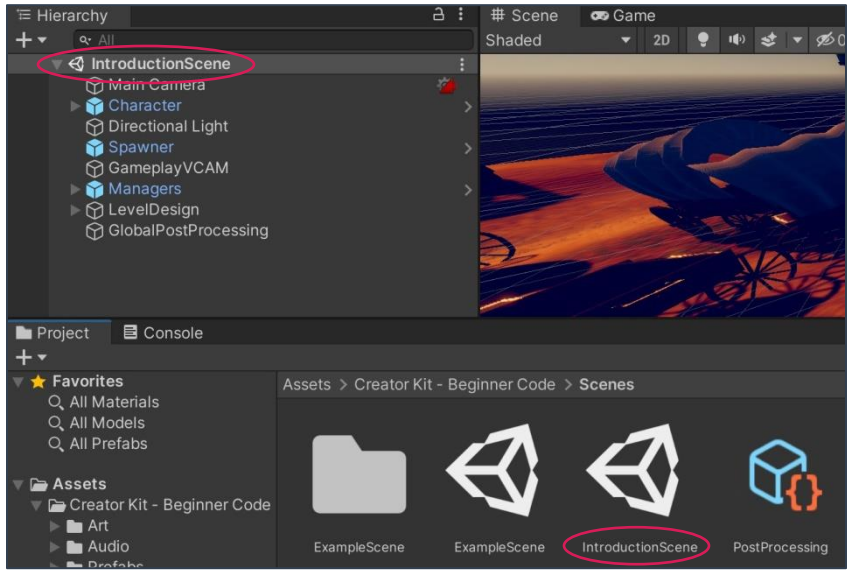
- Most popular game engine IDE in the world.
- Over 50% of smartphone games created with Unity. Maybe, PC games also that.
- It supports 3D/2D programming, physics, lightning, motion capture (for Vtubers) and so on.
- Many 3d models, materials, audio, animations and effects (=assets) can be download.
- C# is supported.
- You can use Unity free!

Understand basics of Unity

To create your game

1. Prepare to edit this game what you want.
    a. Select below folder again.
    “Assets/Creator Kit – Beginner Code/Scenes”
    b. Select “ExampleScene”.
    c. Select Edit -> Duplicate. →
    d. Select “Example scene 1”
    and right click -> “rename” as “IntroductionScene”.
    e. Double click “IntroductionScene”.
    f. Check the below 2 points. →
        - Hierarchy shows “IntroductionScene”.
        - Project shows “IntrodctionScene”.
2. Congratulations! You have became god on this game!
    - Try to use  tools freely.
    - If you want to cancel your last operation, use ctrl+z (windows) or cmd+z (mac).
    - If you want to discard all of your changes, double click “ExampleScene” -> select “Don’t save” and open again.





Understand basics of Unity

To create your game

Components: Gives feature to GameObjects.

e.g. Audio, Render, Physics, Textures, Control Scripts… etc.
Each Component has Properties.

GameObjects: All objects in the game.

e.g. Camera, Light, Buildings, Player character, Enemies, Portions, Ground, Weapons… etc.
Each GameObject has Components.

Scenes: Game project is composed by some of scenes.

e.g. Title Scene, Stage 1 Scene, Game Over Scene… etc.
Each scene has GameObjects.

Understand basics of Unity

To create your game

1. Please translate 5. Review the Unity Editor interface.
2. Please translate 6. Review the Unity Editor toolbar.
3. Attachments to explain keywords on Unity:

Scenes: Game project is composed by some of scenes.
e.g. Title Scene, Stage 1 Scene, Game Over Scene… etc. Each scene has GameObjects.

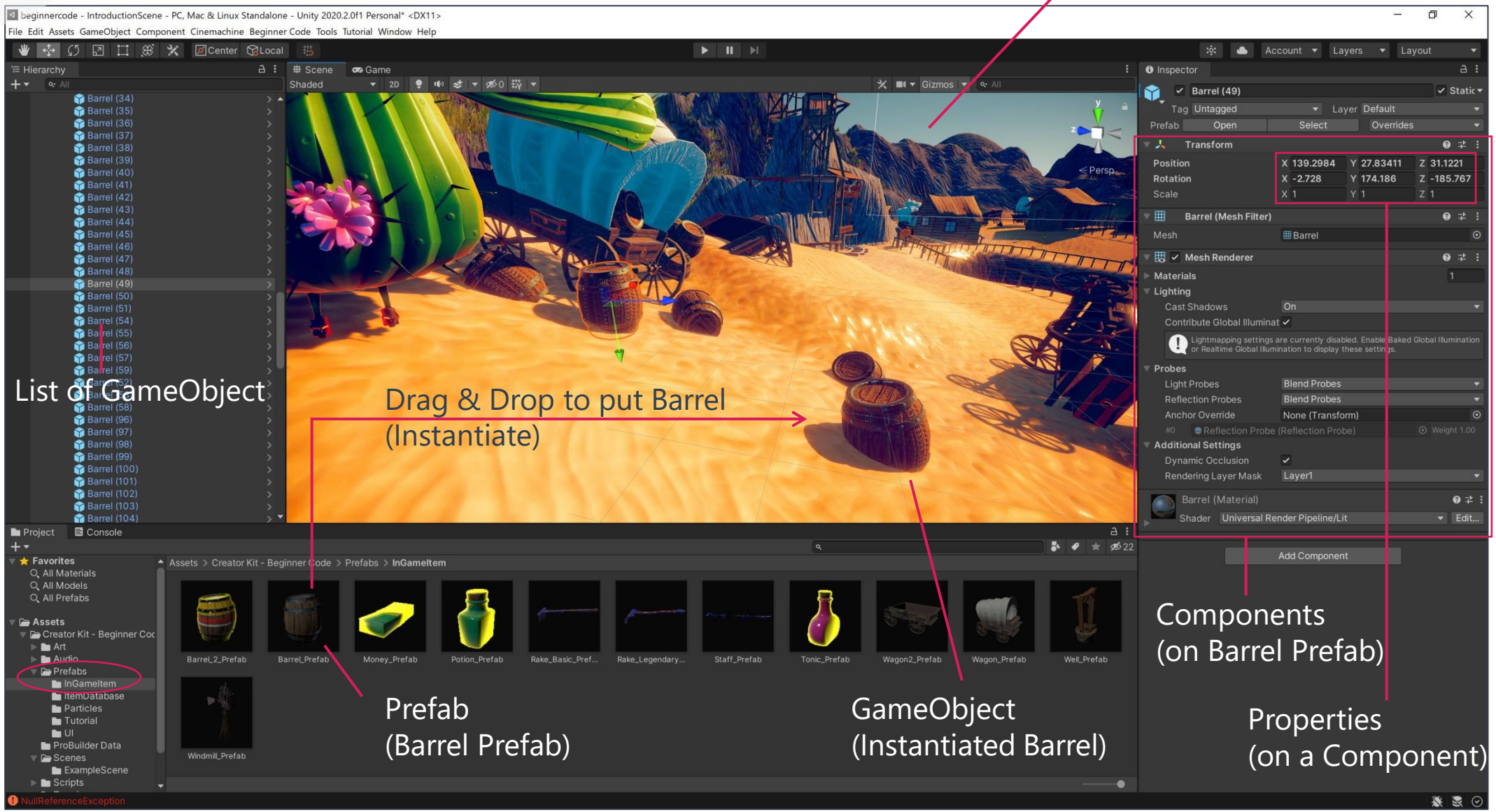
GameObjects: All objects that appear in the game called GameObject.
e.g. Player character, Enemies, Ground, Portions, Weapons… etc. Each GameObject has Components.

Components: It gives feature to GameObjects. A plate GameObject become wall attached by components.
e.g. Physics, Audio, Render, Textures, Control, Scripts… etc. Each Component has Properties.

Prefabs: It is template to create copies of a GameObject with Components. By using wall Prefab, you can put same condition of wall into the Scene. The created copies are called Instance.

Understand basics of Unity

To create your game

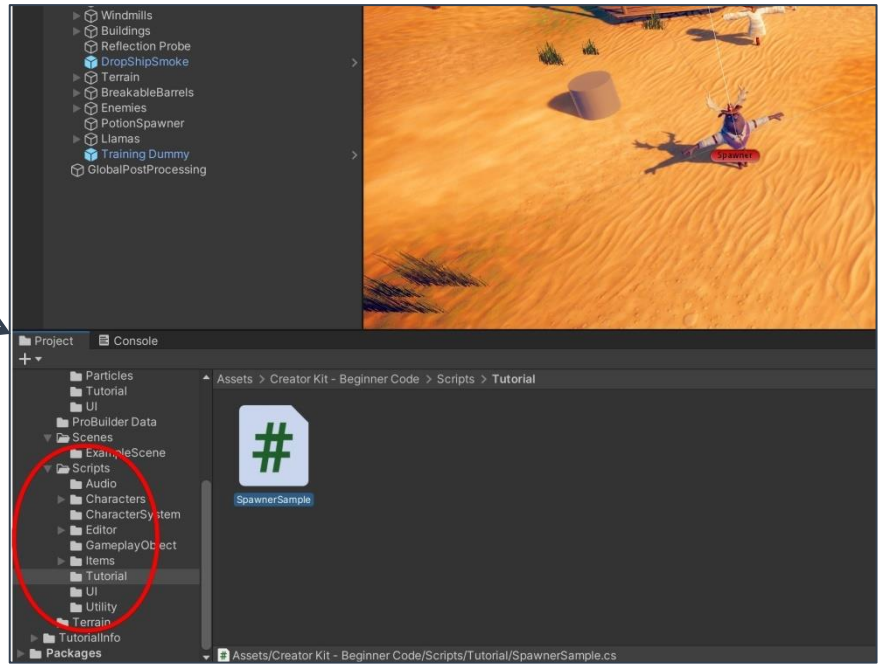


Understand basics of Programming

Architecture of scripts

1. Find “SpawnerSample” script on Project.
2. Click the script.
3. Inspector shows the contents of this script.

   This is program to create 3 portions on the scene.



Spawner Sample (Mono Script)

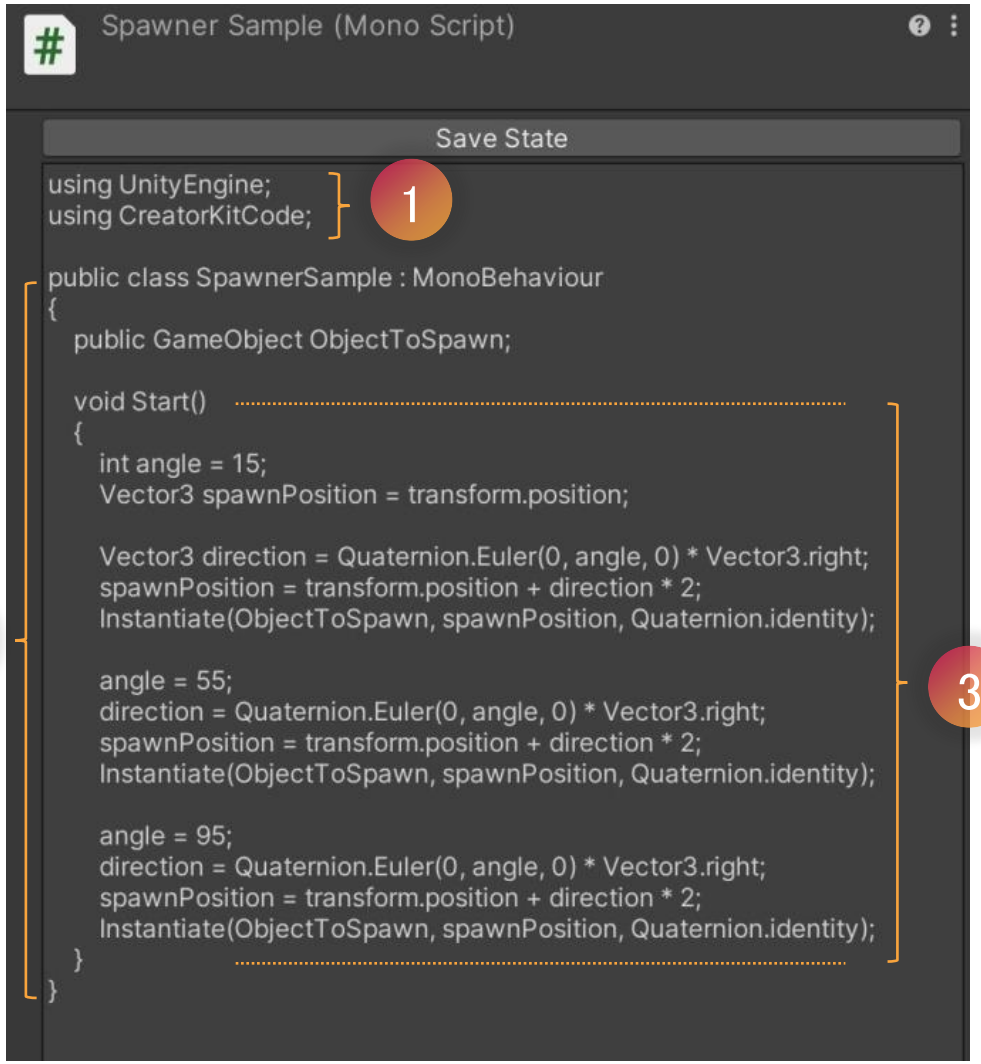
Save State

```
using UnityEngine;
using CreatorKitCode;

public class SpawnerSample : MonoBehaviour
{
    public GameObject ObjectToSpawn;

    void Start()
    {
        int angle = 15;
        Vector3 spawnPosition = transform.position;

        Vector3 direction = Quaternion.Euler(0, angle, 0) * Vector3.right;
        spawnPosition = transform.position + direction * 2;
        Instantiate(ObjectToSpawn, spawnPosition, Quaternion.identity);

        angle = 55;
        direction = Quaternion.Euler(0, angle, 0) * Vector3.right;
        spawnPosition = transform.position + direction * 2;
        Instantiate(ObjectToSpawn, spawnPosition, Quaternion.identity);

        angle = 95;
        direction = Quaternion.Euler(0, angle, 0) * Vector3.right;
        spawnPosition = transform.position + direction * 2;
        Instantiate(ObjectToSpawn, spawnPosition, Quaternion.identity);
    }
}
```

You don’t have to understand this script now, It’s too difficult!
I’ll explain it in order.

Understand basics of Programming

Architecture of scripts



1. Namespace Definitions

- using statements is used to express relationship between this script and the other. These 2 scripts are imported.
- It's called namespace.

2. Class Definitions

- public class statements is used to express the script has folowing class. This script has "SpawnerSample" class.
- "{" and "}" show the class range (scope).

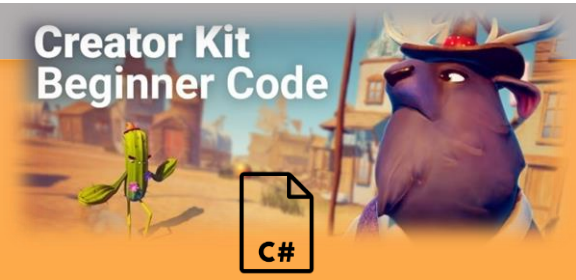
3. Function (Method) Definitions

- void Start() statements is used to express the class has Start function. void and () are described later.
- Functions are also scoped by "{" and "}".

Understand basics of Programming

Architecture of scripts

1. Class is blueprint of scripts. Unity provides a lot of classes.
2. Class can inherit variables and functions of its parent.
3. Namespace provides shortcut to call other classes' factor.



Component class

```
namespace UnityEngine
{
    public class  Component : Object
    {
        public GameObject gameObject;
        public String tag;
        public Transform transform;
        ...

        public Component GetComponent
(string type)
        {
            ...
        }
        ...
    }
}
```

Behaviour class

```
namespace UnityEngine
{
    public class  Behaviour : Component
    {
        public bool enabled;
        public bool isActiveAndEnabled;
        ...
    }
}
```

MonoBehaviour class

```
namespace UnityEngine
{
    public class  MonoBehaviour : Behaviour
    {
        public bool runInEditMode;
        public bool useGUILayout;

        public void Start()
        {
            // In fact, it is defined elsewhere.
        }

        public void Update()
        {
            // In fact, it is defined elsewhere.
        }
        ...
    }
}
```

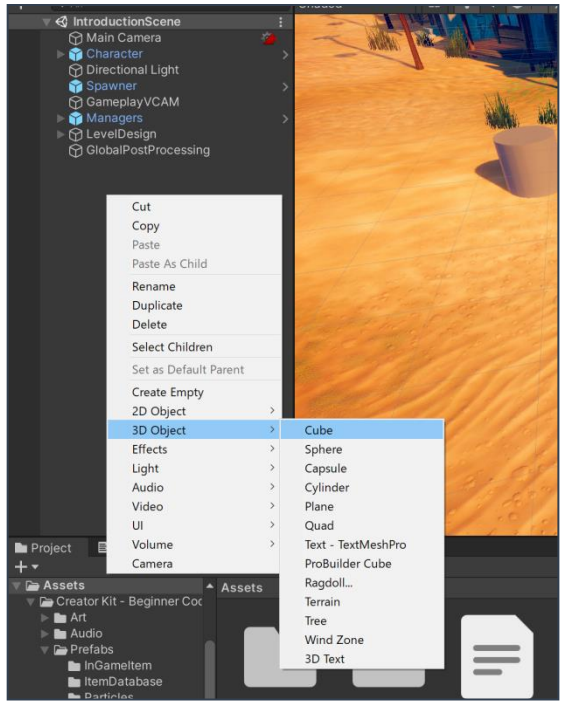
SpawnerSample class

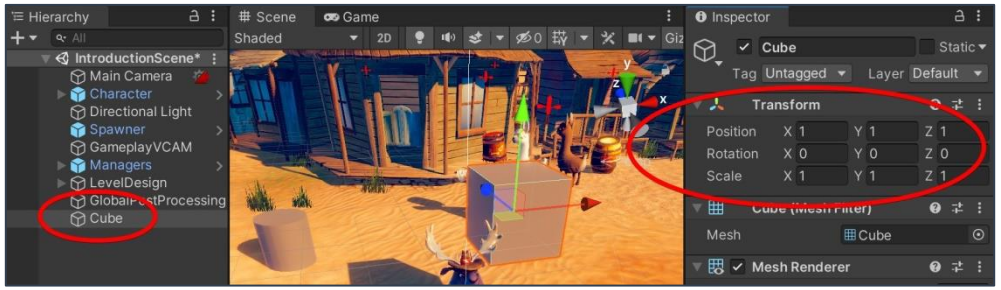
```
using UnityEngine;
using CreatorKitCode;

public class  SpawnerSample : MonoBehaviour
{
    public GameObject ObjectToSpawn;

    public void Start()
    {
        int angle = 15;
        Vector3 spawnPosition = transform.position;
        ...
    }
}
```

Try to write own Script

The first step of programming.

1. Create GameObject to attach script.
    a. Right click blank space at Hierarchy.
    b. Select 3D Object -> Cube.
    c. Select Cube and set Transform.position to (1, 1, 1).
    d. Play the game and check the cube appeared.
2. Add Physics to the cube.
    a. Select Add Component on inspector of Cube.
    b. Select Physics -> Rigidbody.
    c. Play the game and check the changes.
3. Create script file for cube.
    a. Open Scripts/Tutorial folder on project.
    b. Right click blank space at tutorial folder and select Create -> C# Script.
    c. Right click the new one and select Rename to “CubeController”.
    d. Open CubeController (Double click).





Try to write own Script

The first step of programming.

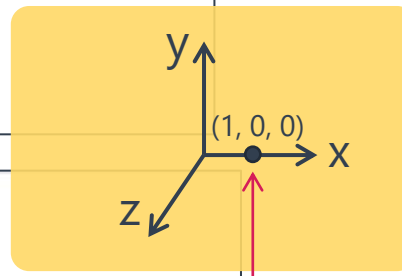
1. Check architecture of the script.
   a. // means 1 line comments.
   b. using means importing class and functions.
   c. /* ~ */ also means multiple line of comments.
   d. public class statement is class definition.
   e. void Start() and void Update() are functions.

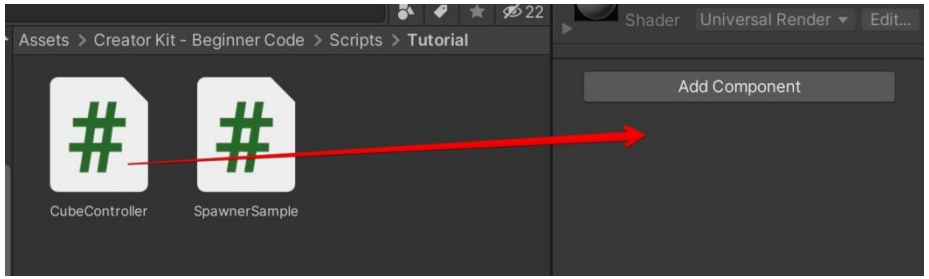
2. Write the script for cube.
   a. Edit Update() method according to this image.
      You don't have to write my comments.
   b. Save (ctrl+S / cmd+s) the file and back to the Unity.

3. Attach the script as Cube component.
   a. Select Cube.
   b. Drag & Drop "CubeController" into Inspector.
      (Near the "Add Component")
   c. Play the game on Unity.

```
1   // Import C# and Unity functions
2   using System.Collections;
3   using System.Collections.Generic;
4   using UnityEngine;
5
6   /*
7    * Check or edit the class name according to file namme
8    *  -> "CubeController"
9    */
    Unity スクリプト | 0 個の参照
10  public class CubeController : MonoBehaviour
11  {
12      // Start is called before the first frame update
        Unity メッセージ | 0 個の参照
13      void Start()
14      {
15
16      }
17
18      // Update is called once per frame
        Unity メッセージ | 0 個の参照
19      void Update()
20      {
21
22      }
23  }
24
```

```
// Update is called once per frame
Unity メッセージ | 0 個の参照
void Update()
{
    /*
     * To move cube, change the transform.position
     * by adding Vector3.right. It means (1,0,0).
     */
    transform.position = transform.position + Vector3.right;
}
```





Try to write own Script

The first step of programming.

1. Fix the cube speed (It's too fast!).
   a. Edit CubeController according to this image. → You don't have to write my comments.
   b. Save the file and back to the Unity.
   c. Play the game again.

```
public class CubeController : MonoBehaviour
{
    /*
     * Definition of speed variable
     * Variable type: float (decimal number)
     */
    public float speed = 0.5f;

    // Start is called before the first frame update
    Unity メッセージ 0 個の参照
    void Start()
    {

    }

    // Update is called once per frame
    Unity メッセージ 0 個の参照
    void Update()
    {
        /*
         * To move cube, change the transform.position
         * by adding Vector3.right. It means (1,0,0).
         */
        transform.position = transform.position + Vector3.right * speed;
    }
```

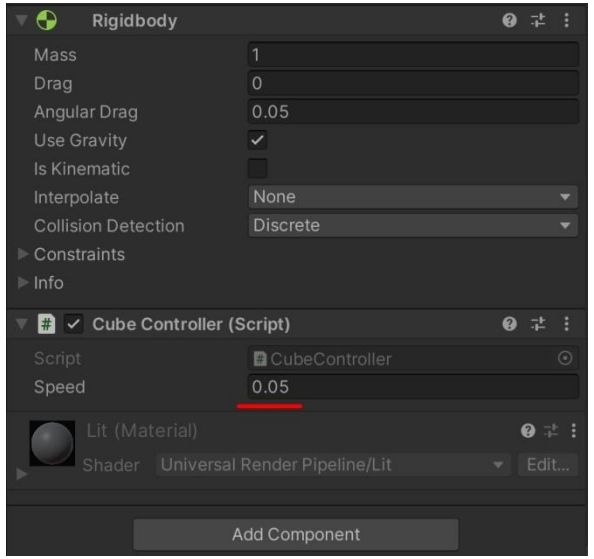
2. Still too fast? OK, let's fix it on Inspector.
   a. Select Cube and find CubeController on Inspector.
   b. Edit the "Speed" to 0.05.
   c. Play the game again.

0.05

Variable

(float) speed

Types	Means	Examples
int	Integer	1
float	Decimal	0.5f
string	Characters	"Unity"
bool	Boolean	true or false
Vector3	x,y,z	new Vector3(1,1,1)
GameObject	Object	Object name



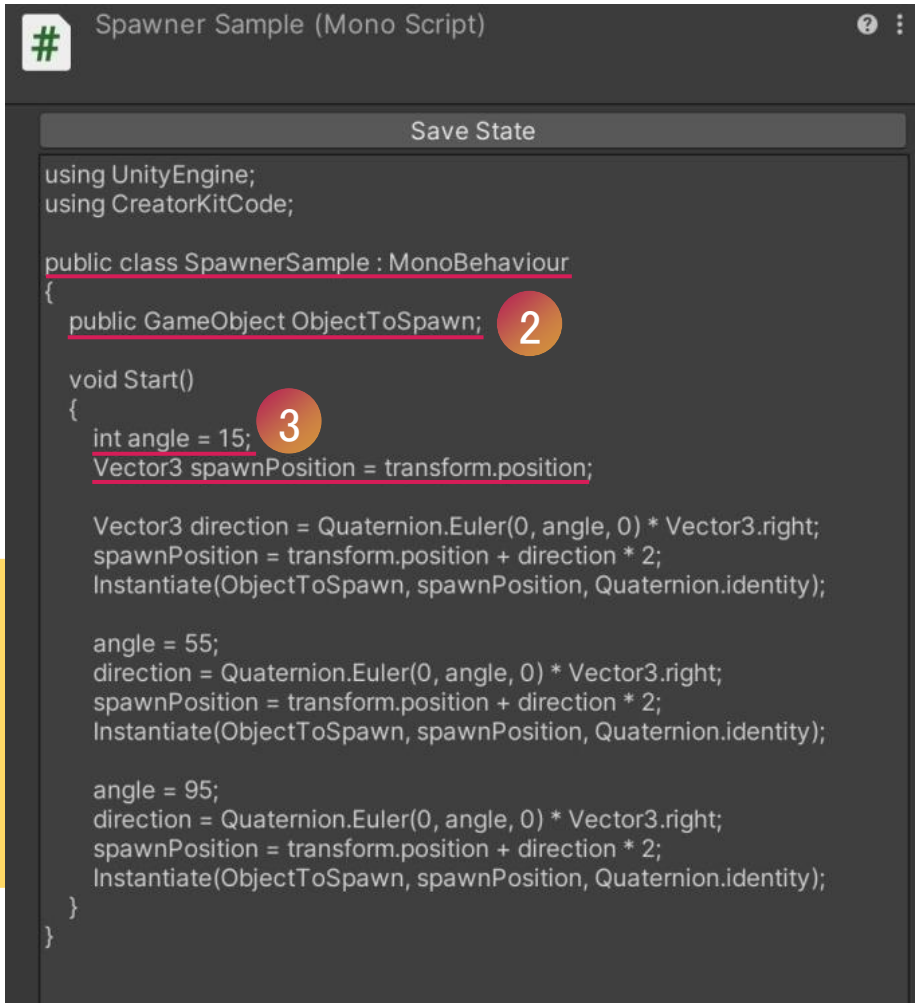
Try to write own Script

The first step of programming.

1. Please translate 2. Find an instruction in the script.
2. Please translate 3. Introduction to variables.
3. Please translate 4. Identify the type and name of the variable.
4. Please translate 5. Identify the value assigned to the variable.
5. You can understand parts of red line on this image.

3 case of variables

1. Global (private) variable: Only used in the class scope.
2. Global public variable: Used in classes scope + Unity inspector.
3. Local variable: Used in the function scope.



Attachment

This is important!

Coding Rules: Examples

Target	Major Rules	Examples
Class name	a. UpperCamel	a. SampleClass
Class variables	a. LowerCamel b. Snake_case c. Underscore	a. sampleVariable b. sample_variable c. _sampleVariable (Only for private variables)
Bool variables	a. Is_bool	a. is_run
Methods	a. UpperCamel	a. SampleMethod()
Constants	a. All-Caps b. UpperCamel	a. MAX_NUMBER b. MaxNum
Scene name	a. UpperCamel	a. TitleScene
Object name	a. UpperCamel	a. CactusNails
If Statement	a. if (target < condition)	a. if (num < 10)
Comments	You don't' need to write all of contents. Good code & Good name explains the role or function itself.	
Scope	Try to minimize the range.	
Hardcoding	Don't use it. You should use constants.	
Core mind	Simple and Dry. Functions should be under 30〜50 lines.	

Target	Pronunciation	Examples
()	Parenthesis	void Method()
[]	Bracket	int array[3]
{}	Brace	if (num < 10) { }

Attachment

This is important!

When you get Errors!

Check Points / ToDo	Details
Spell miss	Check your script carefully! We need to be case-sensitive in programming.
Error messages	Error messages may indicate the cause of the error or the corresponding line.
Search on Google	Type the error message into Google search. You may find the solutions. Sometimes, the errors / cautions can be ignored.
Narrow down the cause	Use comment out or Debug.Log("") wisely to find the problem.
Attached components	Check inspector on the GameObjects. You may forgot attach the components.
Careless miss	• Forgotten to save the scripts • Target GameObject was disabled • Log message was disabled
When you ask it…	When you ask someone the problem, make sure you clarify the followings. • **What problems happened?** • **When the problems happened?** • **Were you able to identify where is problem?** • **How did you try to solve it so far?** • **What is your goal?**

2.Basic programming with Unity

Class Plan

1. Introduction
2. **Hello world in Unity**
3. Basic programming
4. Object oriented programming
5. Interface programming
6. Programming design
7. Programming design 2
8. Improvement and testing 1
9. Improvement and testing 2
10. Summary

2
Hello world in Unity

2. Hello world in Unity

Class summary

1. Explain the variables and data type.
2. Explain basic arithmetic operations.
3. Explain the functions and scope.
4. Explain the conditional statements.
5. Explain debug method.

Review of Previous Class

The first step of programming.

Types	Means	Examples
int	Integer	1
float	Decimal	0.5f
string	Characters	“Unity”
bool	Boolean	true or false
Vector3	x,y,z	new Vector3(1,1,1)
GameObject	Object	Object name

Create CubeController script.

```
1     // Import C# and Unity functions
2     using System.Collections;
3     using System.Collections.Generic;
4     using UnityEngine;
5
6     /*
7      * Check or edit the class name according to file namme
8      *  -> "CubeController"
9      */
      Unity スクリプト|0 個の参照
10    public class CubeController : MonoBehaviour
11    {
12        /*
13         * Definition of speed variable
14         * Variable type: float (decimal number)
15         */
16        public float speed = 0.5f;   2
17
18        // Start is called before the first frame update
          Unity メッセージ|0 個の参照
19        void Start()
20        {
21
22        }
23
24        // Update is called once per frame
          Unity メッセージ|0 個の参照
25        void Update()
26        {
27            /*
28             * To move cube, change the transform.position
29             * by adding Vector3.right. It means (1,0,0).
30             */
31            transform.position = transform.position + Vector3.right * speed;
32        }
33    }
```

3

3 case of variables

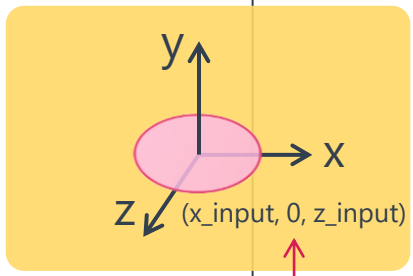
1. Global (private) variable: Only used in the class scope.
2. Global public variable: Used in classes scope + Unity inspector.
3. Local variable: Used in the function scope.

Try to write own Script

The first step of programming.

1. Fix the CubeController.
    a. Edit CubeController according to this image. You don't have to write my comments.
    b. Save the file and back to the Unity.
    c. Play the game again.
        - You can move Cube with arrow keys.

```
Unity スクリプト|0 個の参照
public class CubeController : MonoBehaviour
{
    public float speed = 0.5f;

    // Start is called before the first frame update
    Unity メッセージ|0 個の参照
    void Start()
    {

    }

    // Update is called once per frame
    Unity メッセージ|0 個の参照
    void Update()
    {
        // Variable definition for Input ( ← and → )
        float x_input = Input.GetAxis("Horizontal");

        // Variable definition for Input ( ↑ and ↓)
        float z_input = Input.GetAxis("Vertical");

        // Substitute transform.position(Vector3) + inputed value as Vector3.
        transform.position += new Vector3(x_input,0,z_input) * speed;
    }
}
```



Description.

- Input.GetAxis(string axisName) method by UnityEngine
    - Returns float value between -1 to +1
    - Can be able "Horizontal", "Vertical" and more. User can define it yourself.
- new Vector3(x, y, z) statement
    - Create Vector3 struct to substitute inputted float values for transform.position (Vector3).
    - Vector3 has float x, float y, float z variables.
    - new statement is used to create class/struct.

Arithmetic operators	Means	Examples
=	Substitution	Speed = 0.5f
==	Equal	A == B
+	Addition, Connection	10 + 20 == 2 "ABC" + "DEF" == "ABCDEF"
-	Subtraction	10 – 5 == 5
*	Multiplication	5 * 8 == 40
/	Division	40 / 8 == 5
%	Remainder	5 % 2 == 1
(operator)=	Operate itself	**(A += 10)** == **(A = A + 10)**

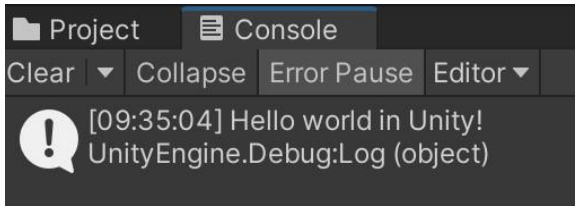
Exercise

Try to basic arithmetic

1. Edit CubeController.Start() according to this image.

```
// Start is called before the first frame update
Unity メッセージ|0 個の参照
void Start()
{
    Debug.Log("Hello world in Unity!");
}
```

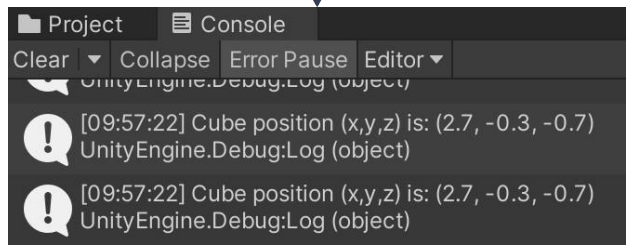
2. Select Console and play the game.
   - Unity shows this message.
   - Debug.log() method shows the message when called it.



3. Edit CubeController.update() according to this image.

```
// Update is called once per frame
Unity メッセージ|0 個の参照
void Update()
{
    // Variable definition for Input ( ← and → )
    float x_input = Input.GetAxis("Horizontal");

    // Variable definition for Input ( ↑ and ↓)
    float z_input = Input.GetAxis("Vertical");

    // Substitute transform.position(Vector3) + inputed value as Vector3
    transform.position += new Vector3(x_input,0,z_input) * speed;
    Debug.Log("Cube position (x,y,z) is: " + transform.position);
}
```

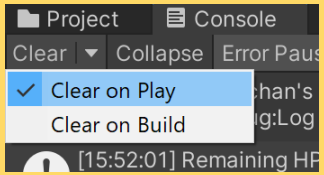
4. Select Console and play the game.
   - Unity shows these messages.
   - Update() is called once per frame, so a lot of messages appeared.
   - Debug.log() method is useful to check parameters.



5. Delete or comment out Debug.Log() on Update().
   - It fills up console by log messages…

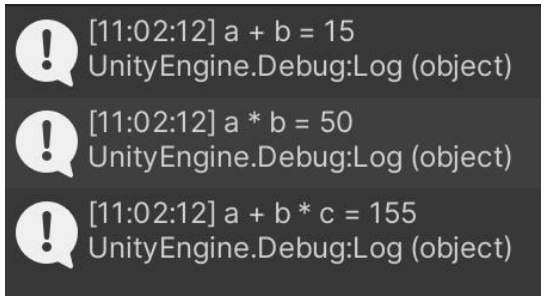
Exercise

Try to basic arithmetic

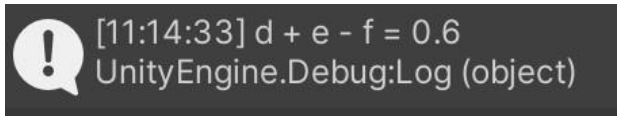


Please use clear option in console.

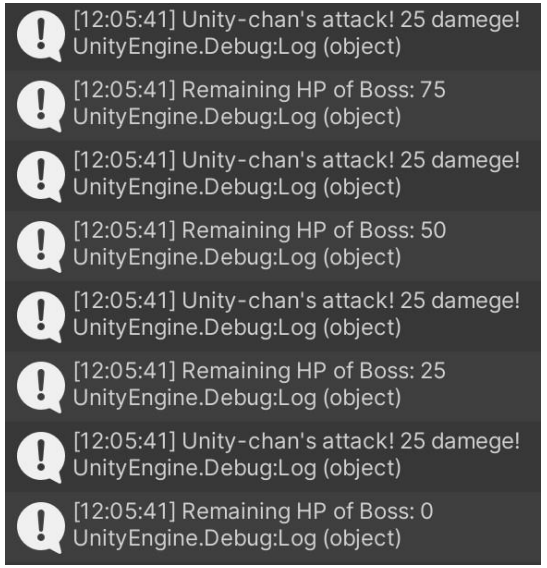
1. Edit Start() to meet below arithmetic.
    a. Define int a = 5, int b = 10, int c = 15.
    b. Calculate and output according to this log on console. →



2. Edit Start() to meet below arithmetic.
    a. Comment out Exercise 1.
    b. Define float d = 1.0, float e = 0.1f, float f = 0.5f, float result;
    c. Calculate {d} + {e} and substitute it for result.
    d. Calculate {result} -= {f} and output it on console. →



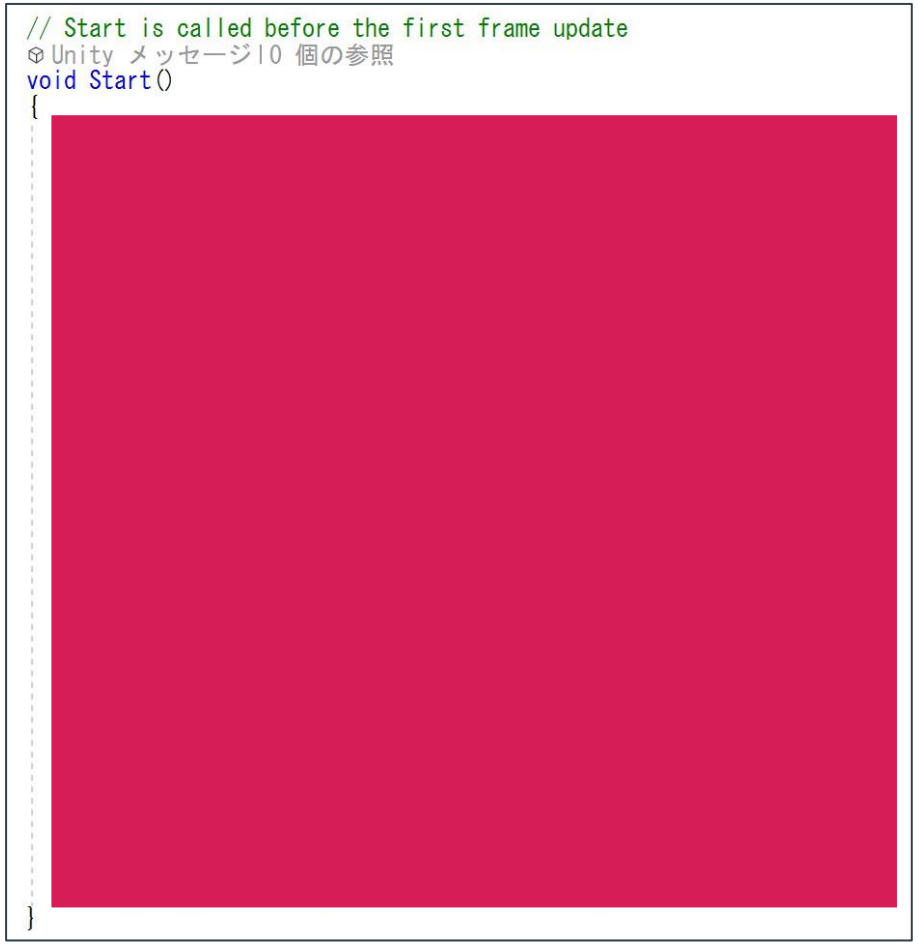
3. Edit Start() to meet below arithmetic.
    a. Comment out Exercise 2.
    b. Define int boss_HP = 100, int player_ATK = 25,
    string player_name = “Unity-chan”.
    c. Calculate {boss_HP} by taking player’s attack with “-=“ operator.
    d. Out put the remaining HP of boss. →
    e. Calculate and output according to this log on console.
    (To beat boss: repeat it until the HP become 0)



Exercise

Try to basic arithmetic

1. Edit Start() to meet below arithmetic.
    a. Define int a = 5, int b = 10, int c = 15.
    b. Calculate and output according to this log on console.
2. Edit Start() to meet below arithmetic.
    a. Comment out Exercise 1.
    b. Define float d = 1.0, float e = 0.1f, float f = 0.5f, float result;
    c. Calculate {d} + {e} and substitute it for result.
    d. Calculate {result} -= {f} and output it on console.
3. Edit Start() to meet below arithmetic.
    a. Comment out Exercise 2.
    b. Define int boss_HP = 100, int player_ATK = 25,
    string player_name = “Unity-chan”.
    c. Calculate {boss_HP} by taking player’s attack with “-=“ operator.
    d. Out put the remaining HP of boss.
    e. Calculate and output according to this log on console.
    (To beat boss: repeat it until the HP become 0)



It’s not Cool...

Try to basic arithmetic

```
// Excercise 3
int boss_HP = 100, player_ATK = 25;
string player_name = "Unity-chan";
boss_HP -= player_ATK;
Debug.Log(player_name + "'s attack! " + player_ATK + " damege!");
Debug.Log("Remaining HP of Boss: " + boss_HP);
boss_HP -= player_ATK;
Debug.Log(player_name + "'s attack! " + player_ATK + " damege!");
Debug.Log("Remaining HP of Boss: " + boss_HP);
boss_HP -= player_ATK;
Debug.Log(player_name + "'s attack! " + player_ATK + " damege!");
Debug.Log("Remaining HP of Boss: " + boss_HP);
boss_HP -= player_ATK;
Debug.Log(player_name + "'s attack! " + player_ATK + " damege!");
Debug.Log("Remaining HP of Boss: " + boss_HP);
}
```

1. Hard-coded “Boss” string

- Boss also want to attack to player. However, this code always show “Boss”.
- This code don’t have versatile. It shoud be replaced to {target_name} variable.

2. Redundant code

- This code has a lot of same statements. It’s low-readability and editibility.
- It should be replaced to function.

Speck of Attack() function:

1. It uses following variables:
   int attacker_name, int attacker_ATK, string target_name, int target_HP
2. It calculates {target_HP} -= {attacker_ATK}
3. It outputs attack message and remaining HP.

Try to write Functions

The second step of programming.

1. Optimize previous code by using functions.
    a. Edit CubeController according to this image. You don't have to write my comments.
    b. Save the file and back to the Unity.
    c. Play the game again.
        - Check output in console.

Description.

- void FunctionName([Type] variable, ...)
    - variable is called "arguments".
    - variable only can be used in the function.
    - variable has data-copy or data-address of the caller's arguments.

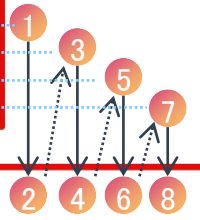
```
Attack(player_name, player_ATK, boss_name, boss_HP);
```

↓ ↓ ↓ ↓

```
void Attack(string attacker_name, int attacker_ATK, string target_name, int target_HP)
```

```
Unity スクリプト | 0 個の参照
public class CubeController : MonoBehaviour
{
    public float speed = 0.5f;

    // Define names as global scope because
    // names will be used in many functions.
    public string player_name = "Unity-chan";
    public string boss_name = "Cuctus-king";

    // Start is called before the first frame update
    Unity メッセージ | 0 個の参照
    void Start()
    {
        // Define variables
        int boss_HP = 100, boss_ATK = 20;
        int player_HP = 85, player_ATK = 25;

        // Call Attack function with variables
        Attack(player_name, player_ATK, boss_name, boss_HP);
        Attack(boss_name, boss_ATK, player_name, player_HP);
        Attack(player_name, player_ATK, boss_name, boss_HP);
        Attack(boss_name, boss_ATK, player_name, player_HP);
    }

    // Define Attack() function with variables
    4 個の参照
    void Attack(string attacker_name, int attacker_ATK, string target_name, int target_HP)
    {
        // Caluculate
        target_HP -= attacker_ATK;

        // Output
        Debug.Log(attacker_name + "'s attack! " + attacker_ATK + " damege!");
        Debug.Log("Remaining HP of " + target_name + ": " + target_HP);
    }
```

Flow of program



Try to write Functions

The second step of programming.

```
[15:52:01] Unity-chan's attack! 25 damege!
UnityEngine.Debug:Log (object)
[15:52:01] Remaining HP of Cuctus-king: 75
UnityEngine.Debug:Log (object)
[15:52:01] Cuctus-king's attack! 20 damege!
UnityEngine.Debug:Log (object)
[15:52:01] Remaining HP of Unity-chan: 65
UnityEngine.Debug:Log (object)
[15:52:01] Unity-chan's attack! 25 damege!
UnityEngine.Debug:Log (object)
[15:52:01] Remaining HP of Cuctus-king: 75
UnityEngine.Debug:Log (object)
[15:52:01] Cuctus-king's attack! 20 damege!
UnityEngine.Debug:Log (object)
[15:52:01] Remaining HP of Unity-chan: 65
UnityEngine.Debug:Log (object)
```

100-25=75

100-25=75

Description.

- void FunctionName([Type] variable, ...)
  - variable is called “arguments”.
  - variable only can be used in the function.
  - variable has data-copy or data-address of the caller’s arguments.

```
Attack(player_name, player_ATK, boss_name, boss_HP);
```

```
void Attack(string attacker_name, int attacker_ATK, string target_name, int target_HP)
```

```
Unity スクリプト | 0 個の参照
public class CubeController : MonoBehaviour
{
    public float speed = 0.5f;

    // Define names as global scope because
    // names will be used in many functions.
    public string player_name = "Unity-chan";
    public string boss_name = "Cuctus-king";

    // Start is called before the first frame update
    Unity メッセージ | 0 個の参照
    void Start()
    {
        // Define variables
        int boss_HP = 100, boss_ATK = 20;
        int player_HP = 85, player_ATK = 25;

        // Call Attack function with variables
        Attack(player_name, player_ATK, boss_name, boss_HP);
        Attack(boss_name, boss_ATK, player_name, player_HP);
        Attack(player_name, player_ATK, boss_name, boss_HP);
        Attack(boss_name, boss_ATK, player_name, player_HP);
    }

    // Define Attack() function with variables
    4 個の参照
    void Attack(string attacker_name, int attacker_ATK, string target_name, int target_HP)
    {
        // Caluculate
        target_HP -= attacker_ATK;

        // Output
        Debug.Log(attacker_name + "'s attack! " + attacker_ATK + " damege!");
        Debug.Log("Remaining HP of " + target_name + ": " + target_HP);
    }
```

Try to write Functions

The second step of programming.

1. Refactor the code.
    a. Edit CubeController according to this image. You don't have to write my comments.
    b. Save the file and back to the Unity.
    c. Play the game again.
        - Check output in console.

```
public class CubeController : MonoBehaviour
{
    public float speed = 0.5f;

    // Define names as global scope because
    // names will be used in many functions.
    public string player_name = "Unity-chan";
    public string boss_name = "Cuctus-king";

    // Start is called before the first frame update
    Unity メッセージ | 0 個の参照
    void Start()
    {
        // Define variables
        int boss_HP = 100, boss_ATK = 20;
        int player_HP = 85, player_ATK = 25;

        // Call Attack function with variables
        boss_HP = Attack(player_name, player_ATK, boss_name, boss_HP);
        player_HP = Attack(boss_name, boss_ATK, player_name, player_HP);
        boss_HP = Attack(player_name, player_ATK, boss_name, boss_HP);
        player_HP = Attack(boss_name, boss_ATK, player_name, player_HP);
    }

    // Define Attack() function with variables
    4 個の参照
    int Attack(string attacker_name, int attacker_ATK, string target_name, int target_HP)
    {
        // Caluculate
        target_HP -= attacker_ATK;

        // Output
        Debug.Log(attacker_name + "'s attack! " + attacker_ATK + " damege!");
        Debug.Log("Remaining HP of " + target_name + ": " + target_HP);

        return target_HP;
    }
```

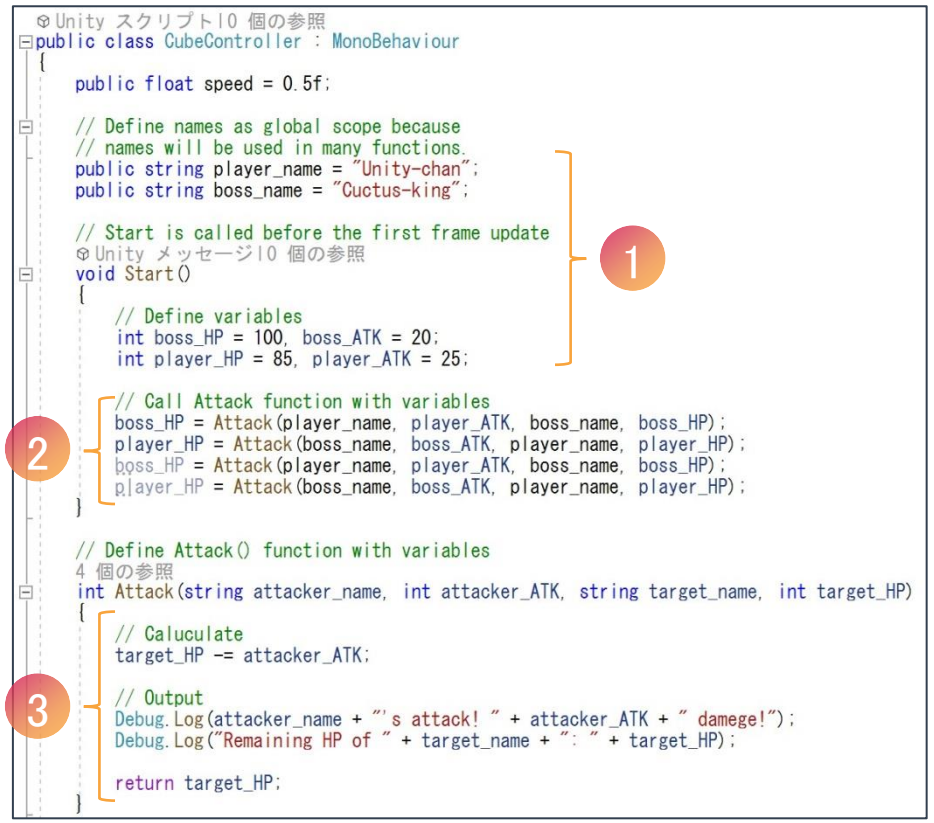
```
[16:36:28] Unity-chan's attack! 25 damege!
UnityEngine.Debug:Log (object)
[16:36:28] Remaining HP of Cuctus-king: 75
UnityEngine.Debug:Log (object)
[16:36:28] Cuctus-king's attack! 20 damege!
UnityEngine.Debug:Log (object)
[16:36:28] Remaining HP of Unity-chan: 65
UnityEngine.Debug:Log (object)
[16:36:28] Unity-chan's attack! 25 damege!
UnityEngine.Debug:Log (object)
[16:36:28] Remaining HP of Cuctus-king: 50
UnityEngine.Debug:Log (object)
[16:36:28] Cuctus-king's attack! 20 damege!
UnityEngine.Debug:Log (object)
[16:36:28] Remaining HP of Unity-chan: 45
UnityEngine.Debug:Log (object)
```

Description.

- [Type] FunctionName([Type] variable, ...)
    - The function returns [Type] value / object.
    - It called "Return value".
    - It needs return statement.

However, still not Cool...

The third step of programming.



```
Unity スクリプト | 0 個の参照
public class CubeController : MonoBehaviour
{
    public float speed = 0.5f;

    // Define names as global scope because
    // names will be used in many functions.
    public string player_name = "Unity-chan";
    public string boss_name = "Cuctus-king";

    // Start is called before the first frame update
    Unity メッセージ | 0 個の参照
    void Start()
    {
        // Define variables
        int boss_HP = 100, boss_ATK = 20;
        int player_HP = 85, player_ATK = 25;

        // Call Attack function with variables
        boss_HP = Attack(player_name, player_ATK, boss_name, boss_HP);
        player_HP = Attack(boss_name, boss_ATK, player_name, player_HP);
        boss_HP = Attack(player_name, player_ATK, boss_name, boss_HP);
        player_HP = Attack(boss_name, boss_ATK, player_name, player_HP);
    }

    // Define Attack() function with variables
    4 個の参照
    int Attack(string attacker_name, int attacker_ATK, string target_name, int target_HP)
    {
        // Caluculate
        target_HP -= attacker_ATK;

        // Output
        Debug.Log(attacker_name + "'s attack! " + attacker_ATK + " damege!");
        Debug.Log("Remaining HP of " + target_name + ": " + target_HP);

        return target_HP;
    }
```

1 2 3

1. Rolls are not divided

- This script have both rolls of player and enemy.
  It's not good because complicated by their parameters.
- To clarify the rolls, we can create Player class and Enemy class. It's called Object-oriented thinking.

2. Manual calling

- Even now, we have to call Attack() repeatedly.
- It's seems good to use loop statement.
  Loop statement is described in next part.

3. Bug can be occurred

- Generally, HP should not become minus number.
- The function have to check target_HP before return statement. It's realized by conditional statements.

Speck change of Attack() function:

1. It check {target_HP} before return it.
2. If {target_HP} < 0, it will be set to 0.
3. It outputs "{target_name} died...".

Try to write If/Else statement

The third step of programming.

1. Fix the cube control.
    a. Edit CubeController according to this image. You don't have to write my comments.
    b. Save the file and back to the Unity.
    c. Play the game and check console.

Description.

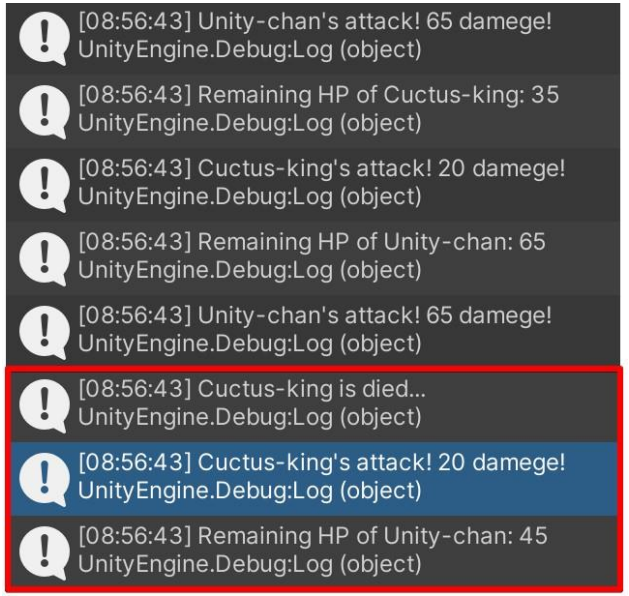
- if (condition) { //programs }
    - Condition becomes true: programs are executed.
    - Condition becomes false: jump to else {}.
- else if (condition2) { //programs }
    - When you need more conditions after if statement, you can use else if statement.
- Conditional operators
    - (target_HP <= 0) : it becomes true when HP is 0
    - (!target_HP >= 0) : it's same as (target_HP < 0) "!" means reverse condition.
    - (target_HP != 0) means not 0. Equal is "==".

```
void Start()
{
    // Define variables
    int boss_HP = 100, boss_ATK = 20;
    int player_HP = 85, player_ATK = 65;

    // Call Attack function with variables
    boss_HP = Attack(player_name, player_ATK, boss_name, boss_HP);
    player_HP = Attack(boss_name, boss_ATK, player_name, player_HP);
    boss_HP = Attack(player_name, player_ATK, boss_name, boss_HP);
    player_HP = Attack(boss_name, boss_ATK, player_name, player_HP);
}

// Define Attack() function with variables
4 個の参照
int Attack(string attacker_name, int attacker_ATK, string target_name, int target_HP)
{
    // Caluculate
    target_HP -= attacker_ATK;

    // Output
    Debug.Log(attacker_name + "'s attack! " + attacker_ATK + " damege!");

    // Check HP
    if (target_HP < 0)
    {
        target_HP = 0;
        Debug.Log(target_name + " is died...");
        return 0;
    }
    else
    {
        Debug.Log("Remaining HP of " + target_name + ": " + target_HP);
        return target_HP;
    }
}
```

Try to write If/Else statement

The third step of programming.

Zombie attack!




```
// Define names as global scope because
// names will be used in many functions.
public string player_name = "Unity-chan";
public string boss_name = "Cuctus-king";

// Start is called before the first frame update
Unity メッセージ|0 個の参照
void Start()
{
    // Define variables
    int boss_HP = 100, boss_ATK = 20;
    int player_HP = 85, player_ATK = 65;

    // Call Attack function with variables
    boss_HP = Attack(player_name, player_ATK, boss_name, boss_HP);
    player_HP = Attack(boss_name, boss_ATK, player_name, player_HP);
    boss_HP = Attack(player_name, player_ATK, boss_name, boss_HP);
    player_HP = Attack(boss_name, boss_ATK, player_name, player_HP);
}

// Define Attack() function with variables
4 個の参照
int Attack(string attacker_name, int attacker_ATK, string target_name, int target_HP)
{
    // Caluculate
    target_HP -= attacker_ATK;

    // Output
    Debug.Log(attacker_name + "'s attack! " + attacker_ATK + " damege!");

    // Check HP
    if (target_HP < 0)
    {
        target_HP = 0;
        Debug.Log(target_name + " is died...");
        return 0;
    }
    else
    {
        Debug.Log("Remaining HP of " + target_name + ": " + target_HP);
        return target_HP;
    }
}
```

Exercise

1. How do you fix the bug with if (??? != ???) in Attack() ?
2. How do you fix the bug by defining boolean flag and if statement?

Try to write If/Else statement

The third step of programming.

```
public class CubeController : MonoBehaviour
```

Exercise

1. How do you fix the bug with if (??? != ???) in Attack() ?
2. How do you fix the bug by defining boolean flag and if statement?

Try to write If/Else statement

The third step of programming.

1. Add rotation key for cube.
    a. Edit CubeController according to this image. You don't have to write my comments.
    b. Save the file and back to the Unity.
    c. Play the game again.
        - Press 'Q' or 'E' key.

```
// Update is called once per frame
Unity メッセージ | 0 個の参照
void Update()
{
    // Variable definition for Input ( ← and → )
    float x_input = Input.GetAxis("Horizontal");

    // Variable definition for Input ( ↑ and ↓)
    float z_input = Input.GetAxis("Vertical");

    // Substitute transform.position(Vector3) + inputed value as Vector3
    transform.position += new Vector3(x_input, 0, z_input) * speed;

    if (Input.GetKey(KeyCode.Q))
    {
        Quaternion rotation = Quaternion.Euler(0, -2, 0);
        this.transform.rotation *= rotation;
    }

    if (Input.GetKey(KeyCode.E))
    {
        Quaternion rotation = Quaternion.Euler(0, 2, 0);
        this.transform.rotation *= rotation;
    }
}
```

Description.

- Input.Getkey()
    - It returns bool (true / false) by user's input.
    - The argument is string or KeyCode enum.
- KeyCode enum
    - It called enumerator to manage constants.
    - KeyCode enum has input related constant. e.g. "Space", "UpArrow" etc.
    - You can access the constants easily.

- Quaternion.Euler(x, y, z)
    - Quaternion is used to rotate objects.
    - Quaternion.Euler(0, -2, 0) means rotate -2 degree on Y axis.
    - "this" means the cube object.

Try to write Functions

The second step of programming.

1. Please translate chapter: Write your own instruction

   1. Find the instruction for potion spawn distance.

   to

   4. Check your script changes.

2. Please translate chapter: Introduction to functions

   1. Challenge: Create another potion at game start.

   to

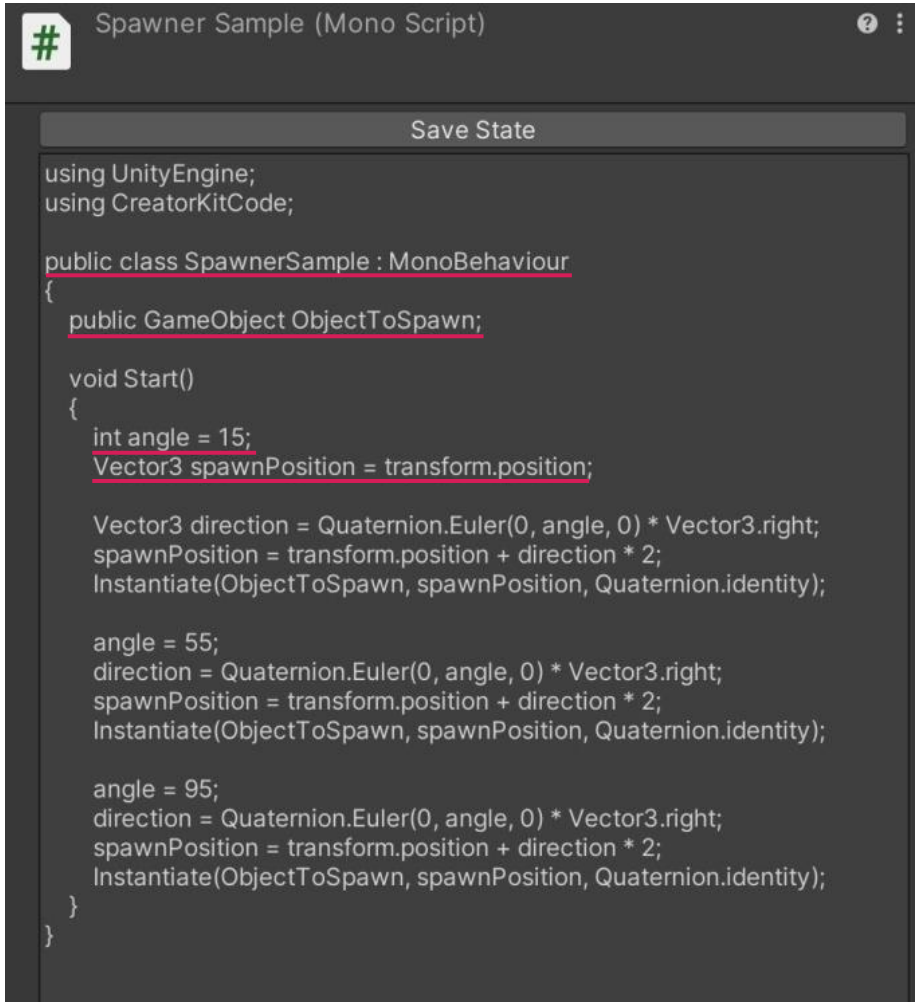
   8. Identify the function that already exists in your script.

3. Please translate chapter: Write your own spawn function

   1. Write a function declaration.

   to

   8. Check your function calls.



Try to write Functions

The second step of programming.


```
Unity スクリプト | 0 個の参照
public class SpawnerSample : MonoBehaviour
{
    public GameObject ObjectToSpawn;

    Unity メッセージ | 0 個の参照
    void Start()
    {
        int angle = 15;
        int radius = 5;
        Vector3 spawnPosition = transform.position;

        Vector3 direction = Quaternion.Euler(0, angle, 0) * Vector3.right;
        spawnPosition = transform.position + direction * radius;
        Instantiate(ObjectToSpawn, spawnPosition, Quaternion.identity);

        angle = 55;
        direction = Quaternion.Euler(0, angle, 0) * Vector3.right;
        spawnPosition = transform.position + direction * radius;
        Instantiate(ObjectToSpawn, spawnPosition, Quaternion.identity);

        angle = 95;
        direction = Quaternion.Euler(0, angle, 0) * Vector3.right;
        spawnPosition = transform.position + direction * radius;
        Instantiate(ObjectToSpawn, spawnPosition, Quaternion.identity);
    }
}
```

```
public class SpawnerSample : MonoBehaviour

    // Attach portion object on inspector.
    public GameObject ObjectToSpawn;

    Unity メッセージ | 0 個の参照
    void Start()
    {
        SpawnPotion(0);
        SpawnPotion(45);
        SpawnPotion(90);
        SpawnPotion(135);
        SpawnPotion(180);
    }

    5 個の参照
    void SpawnPotion(int angle)
    {
        int radius = 5;

        Vector3 direction = Quaternion.Euler(0, angle, 0) * Vector3.right;
        Vector3 spawnPosition = transform.position + direction * radius;

        // Instantiate (Gameobject object, Vector3 position, Quatenion rotation)
        Instantiate(ObjectToSpawn, spawnPosition, Quaternion.identity);
    }
```

Attachment

This is important!

Coding Rules: Examples

Target	Major Rules	Examples
Class name	a. UpperCamel	a. SampleClass
Class variables	a. LowerCamel b. Snake_case c. Underscore	a. sampleVariable b. sample_variable c. _sampleVariable (Only for private variables)
Bool variables	a. Is_bool	a. is_run
Methods	a. UpperCamel	a. SampleMethod()
Constants	a. All-Caps b. UpperCamel	a. MAX_NUMBER b. MaxNum
Scene name	a. UpperCamel	a. TitleScene
Object name	a. UpperCamel	a. CactusNails
If Statement	a. if (target < condition)	a. if (num < 10)
Comments	You don't' need to write all of contents. Good code & Good name explains the role or function itself.	
Scope	Try to minimize the range.	
Hardcoding	Don't use it. You should use constants.	
Core mind	Simple and Dry. Functions should be under 30〜50 lines.	

Target	Pronunciation	Examples
()	Parenthesis	void Method()
[]	Bracket	int array[3]
{}	Brace	if (num < 10) { }

Attachment

This is important!

When you get Errors!

Check Points / ToDo	Details
Spell miss	Check your script carefully! We need to be case-sensitive in programming.
Error messages	Error messages may indicate the cause of the error or the corresponding line.
Search on Google	Type the error message into Google search. You may find the solutions. Sometimes, the errors / cautions can be ignored.
Narrow down the cause	Use comment out or Debug.Log("") wisely to find the problem.
Attached components	Check inspector on the GameObjects. You may forgot attach the components.
Careless miss	• Forgotten to save the scripts • Target GameObject was disabled • Log message was disabled
When you ask it…	When you ask someone the problem, make sure you clarify the followings. • **What problems happened?** • **When the problems happened?** • **Were you able to identify where is problem?** • **How did you try to solve it so far?** • **What is your goal?**

2.Basic programming with Unity

Class Plan

1. Introduction
2. Hello world in Unity
3. **Basic programming**
4. Object oriented programming
5. Interface programming
6. Programming design
7. Programming design 2
8. Improvement and testing 1
9. Improvement and testing 2
10. Summary

3
Basic Programming

3. Basic programming

Class summary

1. Explain Loop statements.
2. Explain Arrays.
3. Explain Enumerators.
4. Improve Usability.
5. Attach Animations.
6. Explain Classes.

Try to write Loop statement

The fourth step of programming.

```
ublic class SpawnerSample : MonoBehaviour

    // Attach portion object on inspector.
    public GameObject ObjectToSpawn;

    Unity メッセージ | 0 個の参照
    void Start()
    {
        SpawnPotion(0);
        SpawnPotion(45);
        SpawnPotion(90);
        SpawnPotion(135);
        SpawnPotion(180);
    }

    5 個の参照
    void SpawnPotion(int angle)
    {
        int radius = 5;

        Vector3 direction = Quaternion.Euler(0, angle, 0) * Vector3.right;
        Vector3 spawnPosition = transform.position + direction * radius;

        // Instantiate (Gameobject object, Vector3 position, Quatenion rotation)
        Instantiate(ObjectToSpawn, spawnPosition, Quaternion.identity);
    }
```

Try to write Loop statement

The fourth step of programming.

1. Optimize the spawn code.
    a. Edit SpawnerSample according to this image. You don't have to write my comments.
    b. Save the file and play the game.

```
Unity メッセージ | 0 個の参照
void Start()
{
    // Repeat procotol from 0 to 4
    for(int i = 0; i < 5; i++)
    {
        // Pass (index * 45) as angle
        SpawnPotion(i * 45);

        // i++ will be executed
        // It means: i = i + 1
    }
}
```

Description: for(A; B; C) statement

- A: Define a variable for loop condition
    - Often used 'i', 'j' and 'k' when we need nested loop.
    - Often set 0 because it's useful when handle arrays.
- B: Loop condition
    - It checked before execute instructions.
    - When true, inside one will be executed.
- C: Prepare for next loop.
    - It executed when this loop reach to end.
    - Often used "i++" (increment) or "i--" (decrement).
    - Be careful to avoid infinite loop. It cause freeze.

```
for(int i = 0; i < 36; i++)
{
    SpawnPotion(i * 10);
}
```

Try to write Loop statement

The fourth step of programming.

1. Expand the spawn code.
    a. Edit SpawnerSample according to this image.
    You don't have to write my comments.
    b. Save the file and play the game.

Description: Array and foreach

- type[] {array_name} = new type[num]
    - Array is container to manage variables.
    - You can access array like array_name[num].
    - It can be nested like if or loop statements.
    It called multidimensional array.
- type[] {array_name} = new type[] {num1, num2…}
    - Declare array with the data.
- foreach(type {variable_name} in {array_name})
    - Access each factor in array to the end.
    - It's useful because we don't have to know the index.
    - You can use break; (exit the loop) and continue;
    (jump to next loop) same as for-loop statement.

```
Unity メッセージ | 0 個の参照
void Start()
{
    // Declare int type Array can contain 5 factors
    int[] angleArray = new int[5];

    // Substitute int values to the Array
    angleArray[0] = 0;
    angleArray[1] = 10;
    angleArray[2] = 50;
    angleArray[3] = 125;
    angleArray[4] = 220;

    // Use all factors of angleArray
    // {array}.Length returns the total number of the factors
    for (int i = 0; i < angleArray.Length; i++)
    {
        // Use i as index of angleArray
        SpawnPotion(angleArray[i]);
    }
}
```

```
Unity メッセージ | 0 個の参照
void Start()
{
    // Define int type Array can contain 5 factors
    int[] angleArray = new int[] {0, 10, 50, 125, 220};

    // Use all factors of angleArray
    foreach(int angle in angleArray)
    {
        SpawnPotion(angle);
    }
}
```

Implement Jump key

To improve your programming skills.

1. Expand the cube function.
    a. Edit CubeController according to this image. You don't have to write my comments.
    b. Save the file and play the game.

```
// Update is called once per frame
Unity メッセージ | 0 個の参照
void Update()
{
    // Variable definition for Input ( ← and → )
    float x_input = Input.GetAxis("Horizontal");

    // Variable definition for Input ( ↑ and ↓)
    float z_input = Input.GetAxis("Vertical");

    // Substitute transform.position(Vector3) + inputed value as Vector3
    transform.position += new Vector3(x_input, 0, z_input) * speed;

    if (Input.GetKey(KeyCode.Q))
    {
        Quaternion rotation = Quaternion.Euler(0, -2, 0);
        this.transform.rotation *= rotation;
    }

    if (Input.GetKey(KeyCode.E))
    {
        Quaternion rotation = Quaternion.Euler(0, 2, 0);
        this.transform.rotation *= rotation;
    }

    if (Input.GetKeyUp(KeyCode.Space))
    {
        this.transform.position += new Vector3(0, 1, 0);
    }
}
```

```
namespace UnityEngine
{
    public enum KeyCode
    {
        None = 0,
        Backspace = 8,
        Tab = 9,
        Clear = 12,
        Return = 13,
        Pause = 19,
        Escape = 27,
        Space = 32,
        Exclaim = 33,
        DoubleQuote = 34,
        Hash = 35,
        Dollar = 36,
        Percent = 37,
        Ampersand = 38,
        Quote = 39,
        LeftParen = 40,
        RightParen = 41,
        Asterisk = 42,
        Plus = 43,
        Comma = 44,
        Minus = 45,
```

Description: enum and GetKeyUp

- enum {enum_name} { }
    - It is container to handle const variables.
    - Often used to create {Days} and {Colors} etc.
    - {Days} enum should have "Monday", "Tuesday"...
    - {Colors} enum should have "Red", "Green", "Blue"...
    - Each element will be set the data as from int 0 by dafault;
    - You can set the data manually.
- Intput.GetKeyUP()
    - This function will be called only once.
    - If you use GetKey(), the cube can be fly while space key has been pressed.

Improve Usability

To improve your programming skills.

3 type of movement

2. Set rigidbody.velocity

- Simple movement with physics. It is located in midlle of real-phisics and game-phisics like double-jump.
- rigidbody.velocity = new Vector3(1, 0, 0) * speed

1. Set transform.position

- Most simple “teleport” movement without physics. It cannot express acceleration.
- transform.position += new Vector3(1, 0, 0)

When use phisics, the instructions sould be written in FixedUpdate() method instead of Update() exept Input.

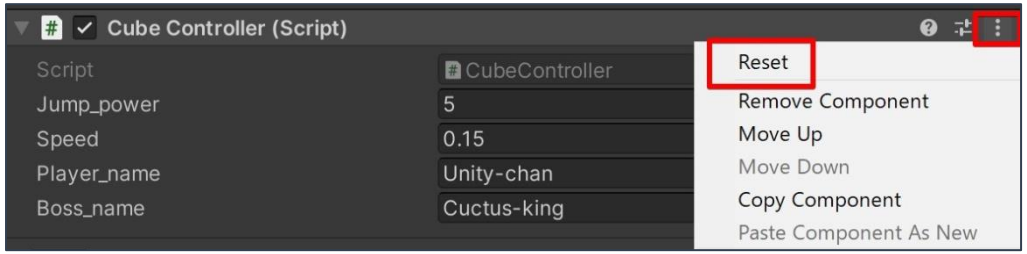
3. Use rigidbody.Addforce()

- Most realistic movement with physics. However, to handle it well, we need knowledge of phisics.
- rigidbody.Addforce(new Vector3(1, 0, 0))

Improve Usability

To improve your programming skills.

1. Improve the jump function.
    a. Edit CubeController according to this image. → You don't have to write my comments.
    b. Save the file.
    c. Select Cube object on Hierarchy.
    d. Select "Reset" to adapt variable settings.



    e. Play the game.

FixedUpdate() method

- It called before caluclate phisics.
- It called constantly regardless of the frame rate so it's not suitable for Input related instructions.

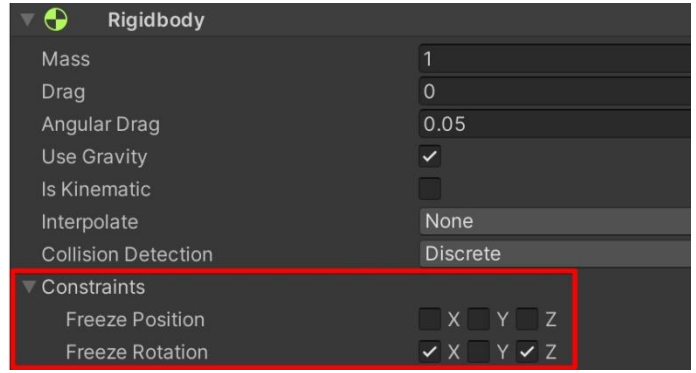
```
public class CubeController : MonoBehaviour
{
    // Declear Rigidbody variable
    Rigidbody rb = null;

    public float jump_power = 5.0f;
    public float speed = 0.15f;

    // Define names as global scope because
    // names will be used in many functions.
    public string player_name = "Unity-chan";
    public string boss_name = "Cuctus-king";
```

```
    void Start()
    {
        // Set Rigidbody to rb
        rb = this.GetComponent<Rigidbody>();

        // Define variables
        int boss_HP = 100, boss_ATK = 20;
        int player_HP = 85, player_ATK = 65;
```

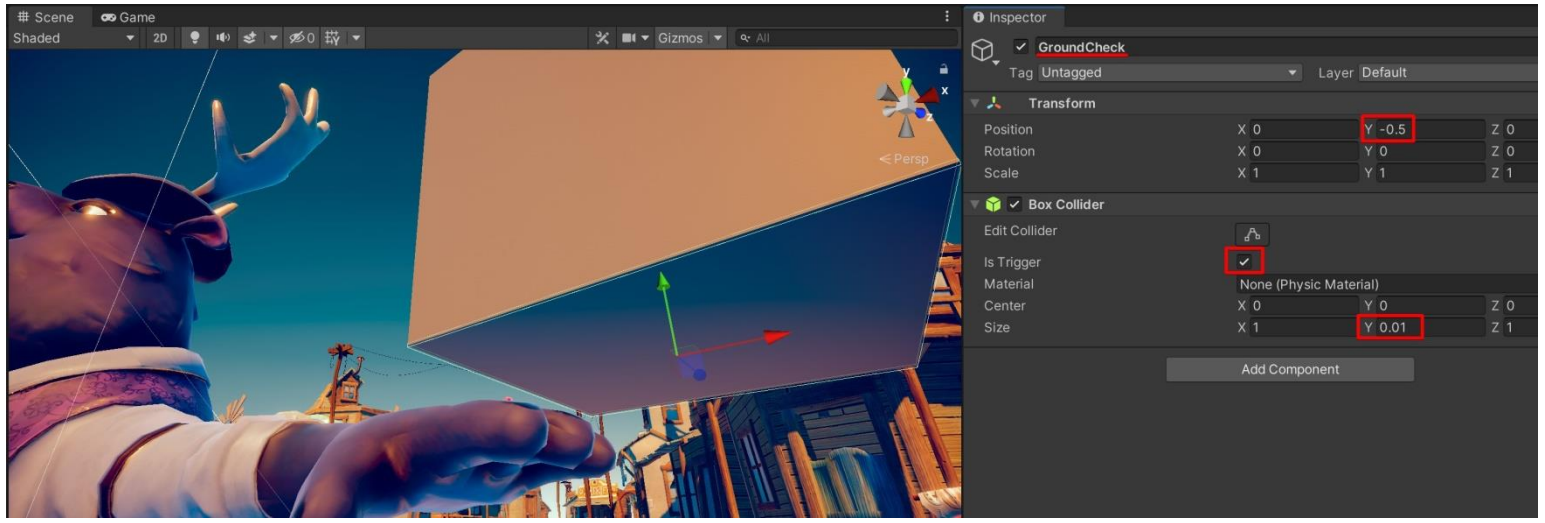
```
void Update()
{
    // Variable definition for Input ( ← and → )
    float x_input = Input.GetAxis("Horizontal");

    // Variable definition for Input ( ↑ and ↓)
    float z_input = Input.GetAxis("Vertical");

    // Substitute transform.position(Vector3) + inputed value as Vector3
    transform.position += new Vector3(x_input, 0, z_input) * speed;

    if (Input.GetKey(KeyCode.Q))
    {
        Quaternion rotation = Quaternion.Euler(0, -2, 0);
        this.transform.rotation *= rotation;
    }

    if (Input.GetKey(KeyCode.E))
    {
        Quaternion rotation = Quaternion.Euler(0, 2, 0);
        this.transform.rotation *= rotation;
    }

    if (Input.GetKeyUp(KeyCode.Space))
    {
        rb.velocity = new Vector3(0, jump_power, 0);
    }
}
```

Improve Usability

To improve your programming skills.



1. Prevent jump related glitches (Infinite jump).
    a. Prevent tumble by adding constraints on X and Z axis. ⟶
    Select Cube object and Check “Freeze Rotation” on Rigidbody.
    b. Create Empty object as “GroundCheck” below Cube object.
    (Right click Cube -> “Create Empty” -> Rename it)
    c. Attach Box Collider to GroundCheck.
    (Select GroundCheck -> Add Component -> Box Collider)
    d. Edit GroundCheck’s position and collider size.
    Move it to bottom of the Cube like thin plane and check “Is Trigger”.



Collider

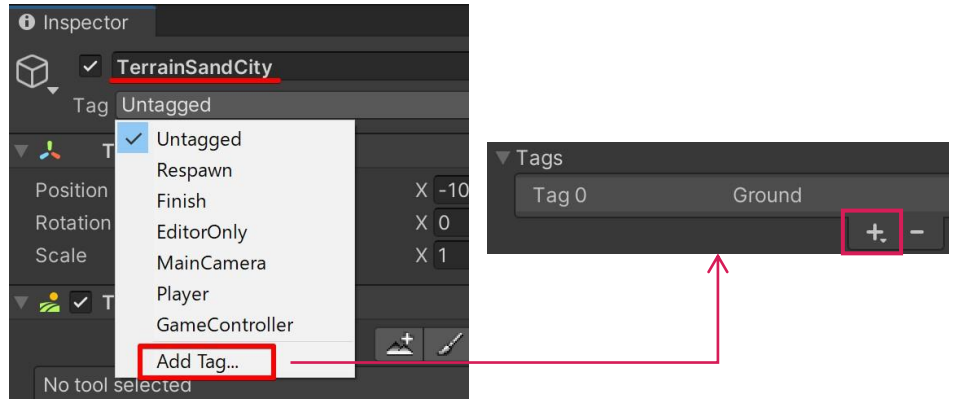
- Component for hit detection.
- When “Is Trigger” is checked, it’s not collide with other objects.

Improve Usability

To improve your programming skills.

1. Prevent jump related glitches (Infinite jump).
    e. Edit CubeController according to this image.
    You don't have to write my comments.
    f. Save the file and play the game to check console logs.
    g. Select Ground object named "TerrainSandCity".
    h. Select "Add Tag" -> + button
    -> Input Tag name as "Ground" on Inspector.
    i. Select Ground object named "TerrainSandCity" again.
    j. Attach "Ground" tag to "TerrainSandCity".
    k. Edit CubeController according to this image.
    You should delete or comment out
    OnTriggerStay() because it dump a lot of logs.
    l. Play the game and check console logs.

```
// When the Cube collide with something
Unity メッセージ | 0 個の参照
private void OnTriggerEnter(Collider other)
{
    Debug.Log("Enter!");
}

Unity メッセージ | 0 個の参照
private void OnTriggerStay(Collider other)
{
    Debug.Log("Stay!");
}

Unity メッセージ | 0 個の参照
private void OnTriggerExit(Collider other)
{
    Debug.Log("Exit!");
}
```




```
Unity メッセージ | 0 個の参照
private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("Ground"))
    {
        Debug.Log("Enter the Ground!");
    }
}
```

OnTriggerEnter() method

- It called when the object collide with other colliders.
- Variable: other has collided object's collider.
- You can check the collider tag with CompareTag().

Improve Usability

To improve your programming skills.

1. Prevent jump related glitches (Infinite jump).
    m. Edit CubeController according to these images. →
    n. Save the file and play the game to check console logs.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CubeController : MonoBehaviour
{
    Rigidbody rb = null;
    public float jump_power = 5.0f;
    public float speed = 0.15f;
    bool is_ground = false;

    void Start()
    {
        rb = this.GetComponent<Rigidbody>();
    }

    void Update()
    {
        float x_input = Input.GetAxis("Horizontal");
        float z_input = Input.GetAxis("Vertical");
        transform.position += new Vector3(x_input, 0, z_input) * speed;

        if (Input.GetKey(KeyCode.Q))
        {
            Quaternion rotation = Quaternion.Euler(0, -2, 0);
            this.transform.rotation *= rotation;
        }

        if (Input.GetKey(KeyCode.E))
        {
            Quaternion rotation = Quaternion.Euler(0, 2, 0);
            this.transform.rotation *= rotation;
        }

        if (Input.GetKeyUp(KeyCode.Space) && is_ground)
        {
            rb.velocity = new Vector3(0, jump_power, 0);
        }
    }
```

```
    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Ground"))
        {
            Debug.Log("Enter the Ground!");
            is_ground = true;
        }
    }

    private void OnTriggerExit(Collider other)
    {
        if (other.CompareTag("Ground"))
        {
            Debug.Log("Exit the Ground!");
            is_ground = false;
        }
    }

}
```

Current source code
without Attack() related things.

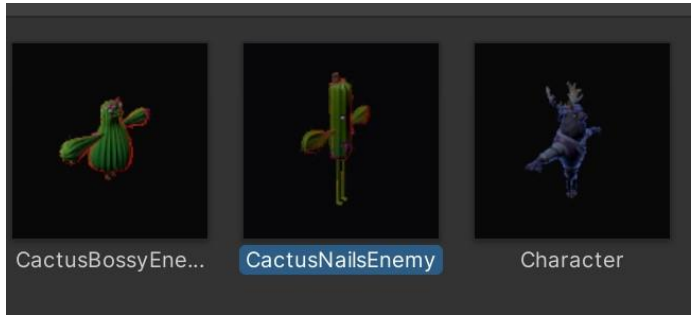
```
Unity スクリプト|0 個の参照
public class CubeController : MonoBehaviour
{
    Rigidbody rb = null;

    public float jump_power = 5.0f;
    public float speed = 0.15f;

    public string player_name = "Unity-chan";
    public string boss_name = "Cuctus-king";

    bool dead_flag = false;
    bool is_ground = false;
```

```
void Update()
{
    // Variable definition for Input ( ← and → )
    float x_input = Input.GetAxis("Horizontal");

    // Variable definition for Input ( ↑ and ↓)
    float z_input = Input.GetAxis("Vertical");

    transform.position += new Vector3(x_input, 0, z_input) * speed;

    if (Input.GetKey(KeyCode.Q))
    {
        Quaternion rotation = Quaternion.Euler(0, -2, 0);
        this.transform.rotation *= rotation;
    }

    if (Input.GetKey(KeyCode.E))
    {
        Quaternion rotation = Quaternion.Euler(0, 2, 0);
        this.transform.rotation *= rotation;
    }

    if (Input.GetKeyUp(KeyCode.Space) && is_ground)
    {
        rb.velocity = new Vector3(0, jump_power, 0);
    }
}

Unity メッセージ|0 個の参照
private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("Ground"))
    {
        Debug.Log("Enter the Ground!");
        is_ground = true;
    }
}

Unity メッセージ|0 個の参照
private void OnTriggerExit(Collider other)
{
    if (other.CompareTag("Ground"))
    {
        Debug.Log("Exit the Ground!");
        is_ground = false;
    }
}
```

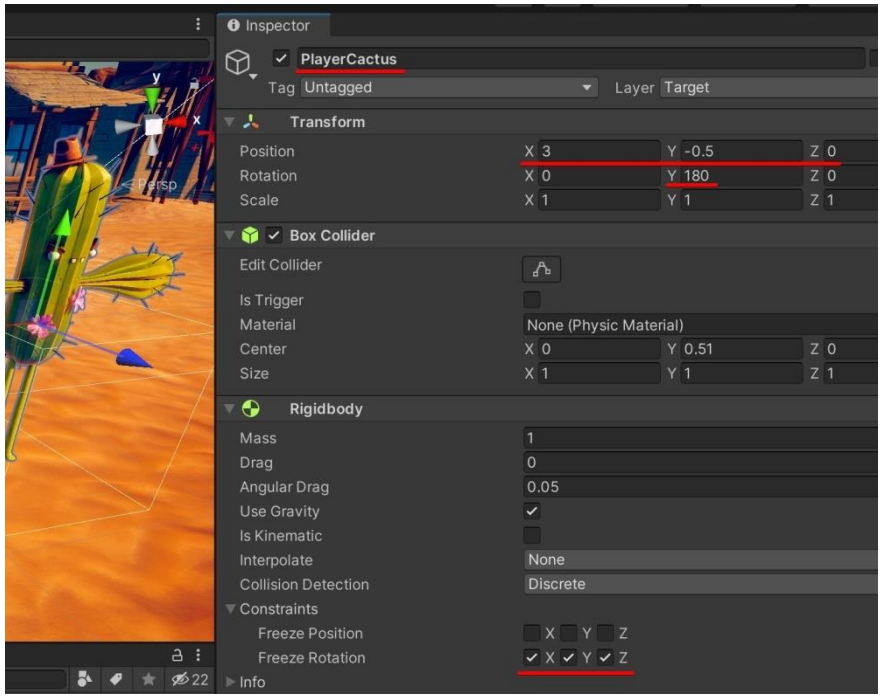
Update Cube into Cactus

Exercise

1. Drag & Drop a prefab of “CactusNailsEnemy” into Hierarchy. ⟶



2. Right click “CactusNailsEnemy” -> Prefab -> Unpack Completely.
3. Rename it to “PlayerCactus”.
4. Remove below components with top-right ⋮ button on PlayerCactus.
    - Nav Mesh Agent
    - Simple Enemy Controller
    - Character Data
    - Character Audio
5. Add Rigidbody and set parameters according to this image. ⟶



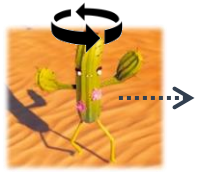
6. Create “CactusController” script on script/tutorial folder.
7. Edit CactusController script according to CubeController.

   You don’t need to write Attack() related code. Use copy & paste.
8. Drag & Drop CactusController into PlayerCactus as component.
9. Play the game and check movement of Cactus.

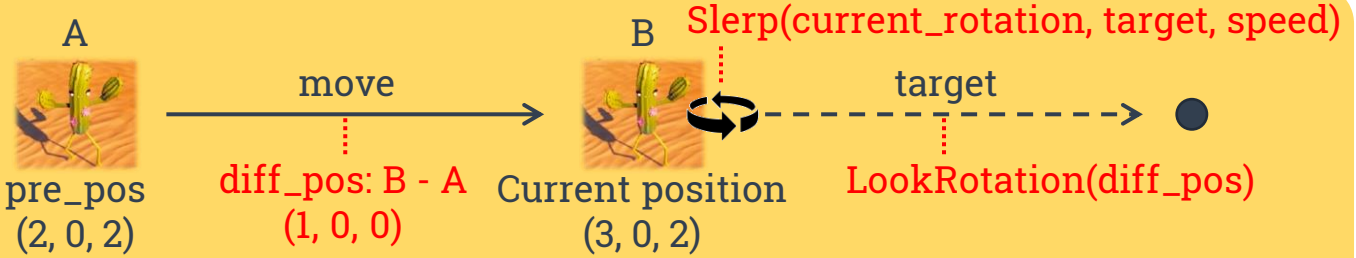
Update Cube into Cactus

Exercise

10. Attach GroundCheck as Empty Object on Hierarchy under CactusNails.
11. Attach Box collider to GroundCheck and move & resize it to the bottom.
12. Check “Is Trigger” on GroundCheck’s Box collider.
13. Play the game and check movement & jump of Cactus.
    - It needs rotation to look forward when you walk.



14. Edit CactusController according to this image and play the game. →



Quaternion.LookRotation(Vector3 forward) :

- Returns Quaternion (≒ rotation) to forward position.

Quaternion.Slerp(Quaternion a, Quaternion b, float t) :

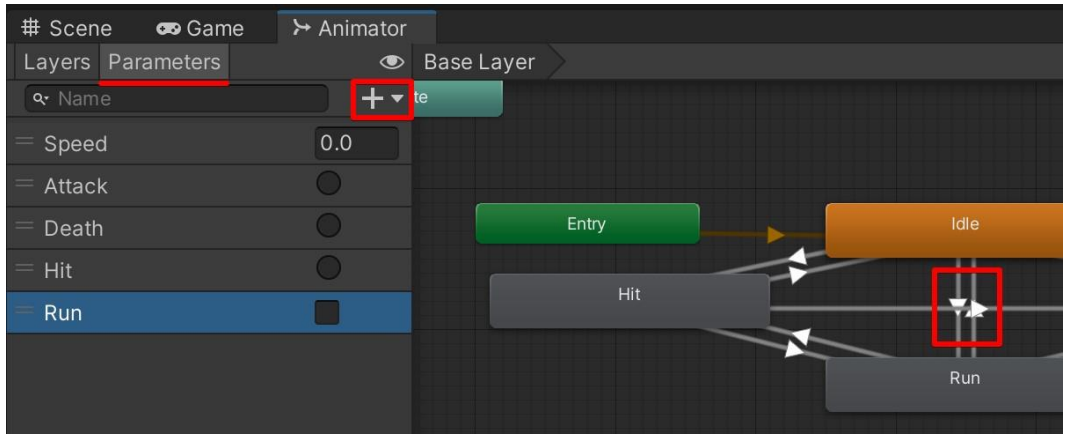
- Returns Quat. as result of spherically interpolates between a and b by t.

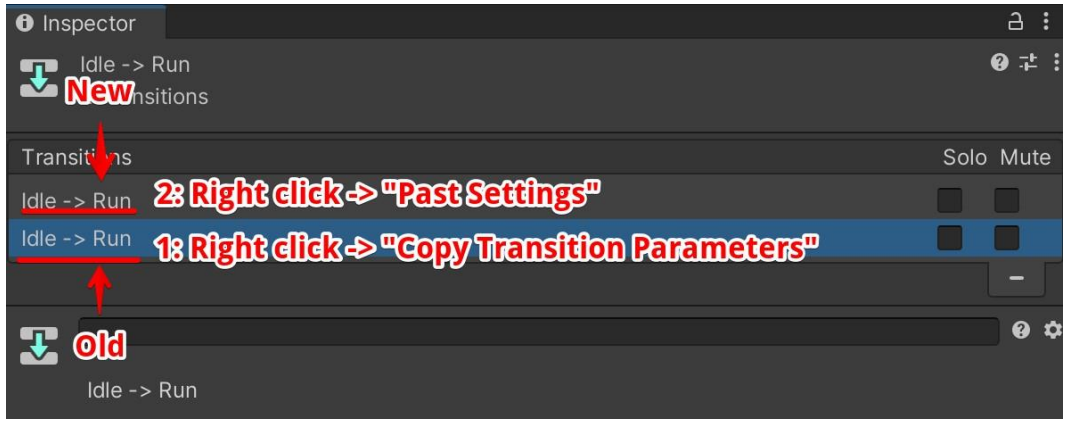
```
Unity スクリプト|0 個の参照
public class CactusController : MonoBehaviour
{
    Rigidbody rb = null;

    public float jump_power = 5.0f;
    public float speed = 0.15f;

    bool is_ground = false;
    Vector3 pre_pos;

    // Start is called before the first frame update
    Unity メッセージ|0 個の参照
    void Start()
    {
        // Set Rigidbody to rb
        rb = this.GetComponent<Rigidbody>();

        // Set Start position
        pre_pos = this.transform.position;
    }

    // Update is called once per frame
    Unity メッセージ|0 個の参照
    void Update()
    {
        // Set current position
        pre_pos = this.transform.position;

        // Variable definition for Input ( ← and → )
        float x_input = Input.GetAxis("Horizontal");

        // Variable definition for Input ( ↑ and ↓)
        float z_input = Input.GetAxis("Vertical");

        transform.position += new Vector3(x_input, 0, z_input) * speed;

        // Calculate difference of position from previous it.
        Vector3 diff_pos = this.transform.position - pre_pos;

        if (diff_pos.magnitude > 0.01f)
        {
            Quaternion target = Quaternion.LookRotation(diff_pos);
            this.transform.rotation =
                Quaternion.Slerp(this.transform.rotation, target, speed);
        }
```

Attach Animations

Exercise

1. Open Animator view by double click “Art/Animations/CactusNailsController” on Project.
2. Select “Parameters” and + button.
   Create “Run” parameter as bool type.
3. Select ↓ to set transition trigger on Idle -> Run.
   However, It already has Speed trigger… Set more way.
   Right click “Idle” statement -> “Make Transition”.
   Then, select “Run” statement to add new transition.
4. Select updated ↓ and copy & paste the setting from old to new one (Upper one) according this image.
5. Select + button on Conditions.
   Add Run condition as true.





Attach Animations

Exercise

7. Set a transition from Run to Idle too.
   Right click “Run” statement -> “Make Transition”.
   Then, select “Idle” statement to add new transition.
8. Select updated ↕ and copy & paste the setting from old to new one (Upper one) according this image.
9. Select + button on Conditions.
   Add Run condition as false.
10. Edit CactusController according to this image. ⟶
11. Play the game. (You can delete Cude)

transform.Find(string name) :

- Returns the transform of child’s objects by the name.

Vector3.magnitude:

- Returns the Vector3’s length: root of ($x^2 + y^2 + z^2$).

```
Unity スクリプト|0 個の参照
public class CactusController : MonoBehaviour
{
    Rigidbody rb = null;
    Animator anim = null;

    public float jump_power = 5.0f;
    public float speed = 0.15f;

    bool is_ground = false;
    Vector3 pre_pos;

    // Start is called before the first frame update
    Unity メッセージ|0 個の参照
    void Start()
    {
        // Set Rigidbody to rb
        rb = this.GetComponent<Rigidbody>();

        // Set Start position
        pre_pos = this.transform.position;

        anim = this.transform.Find("CactusNails").gameObject.GetComponent<Animator>();
    }

    // Update is called once per frame
    Unity メッセージ|0 個の参照
    void Update()
    {
        // Set current position
        pre_pos = this.transform.position;

        // Variable definition for Input ( ← and → )
        float x_input = Input.GetAxis("Horizontal");

        // Variable definition for Input ( ↑ and ↓)
        float z_input = Input.GetAxis("Vertical");

        transform.position += new Vector3(x_input, 0, z_input) * speed;

        // Calculate difference of position from previous it.
        Vector3 diff_pos = this.transform.position - pre_pos;

        if (diff_pos.magnitude > 0.01f)
        {
            Quaternion target = Quaternion.LookRotation(diff_pos);
            this.transform.rotation =
                Quaternion.Slerp(this.transform.rotation, target, speed);
            anim.SetBool("Run", true);
        } else
        {
            anim.SetBool("Run", false);
        }
```

Understand basic of Class

Class as object

- Class is blueprint of scripts.
  -> Declare and implement the factors.
  If declare only, use abstract syntax.
- Class can inherit variables and functions from the parent.
  -> When inherited factors are implemented on the parent class, it can be used without implement in child class. e.g. Move()

Class Variables / Member variables / Fields:

- string name;
- int HP, ATK, SPD;

Class Functions / Methods:

- void Attack()
- void Die()
- void Move() { … // Instructions … }

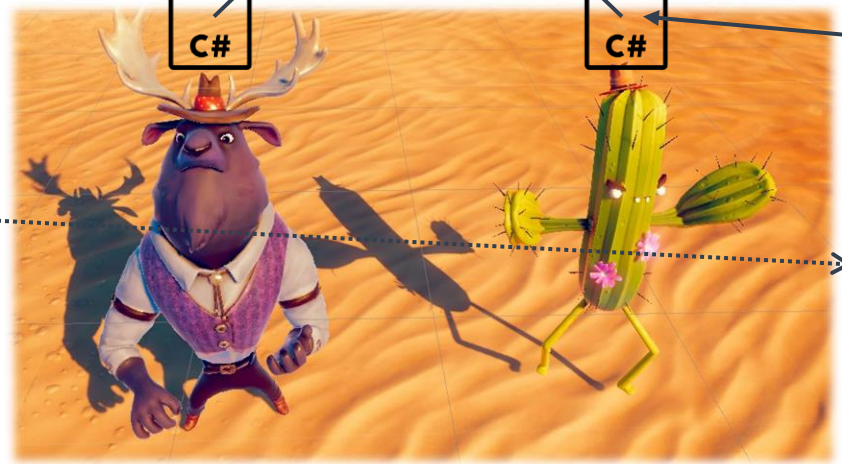
Character class

Constructor

- Method with same name as the class name.
- It's used to set default parameters.

name = MikeMoose;
HP = 20;
ATK = 5;
SPD = 5;

Player(){}
void Attack() {}
void Die() {}
~~void Move(){}~~
void GetItem(){}



Player class

EnemyA class

name = CactusNails;
HP = 11;
ATK = 2;
SPD = 3;
EXP = 2;

EnemyA(){}
void Attack() {}
void Die() {}
~~void Move(){}~~
void DropItem(){}

EnemyABoss class

….
HP = 50;
ATK = 8;
SPD = 1;
EXP = 25;

…

Try to write Class

Architecture of script.

1. Please translate chapter: Introduction to classes

   2. Identify the level above functions

   to

   6. Using class as a type

2. Please translate chapter: Write your own spawner class

   1. Using classes to make your code more efficient

   to

   12. Test your changes

3. Please translate chapter: Introduction to the public keyword and inheritance

   1. Understand the difference between public and private

   to

   11. What does the MonoBehaviour class contain?

Coding Rules: Examples

Target	Major Rules	Examples
Class name	a. UpperCamel	a. SampleClass
Class variables	a. LowerCamel b. Snake_case c. Underscore	a. sampleVariable b. sample_variable c. _sampleVariable (Only for private variables)
Bool variables	a. Is_bool	a. is_run
Methods	a. UpperCamel	a. SampleMethod()
Constants	a. All-Caps b. UpperCamel	a. MAX_NUMBER b. MaxNum
Scene name	a. UpperCamel	a. TitleScene
Object name	a. UpperCamel	a. CactusNails
If Statement	a. if (target < condition)	a. if (num < 10)
Comments	You don't' need to write all of contents. Good code & Good name explains the role or function itself.	
Scope	Try to minimize the range.	
Hardcoding	Don't use it. You should use constants.	
Core mind	Simple and Dry. Functions should be under 30～50 lines.	

Attachment

This is important!

When you get Errors!

Check Points / ToDo	Details
Spell miss	Check your script carefully! We need to be case-sensitive in programming.
Error messages	Error messages may indicate the cause of the error or the corresponding line.
Search on Google	Type the error message into Google search. You may find the solutions. Sometimes, the errors / cautions can be ignored.
Narrow down the cause	Use comment out or Debug.Log("") wisely to find the problem.
Attached components	Check inspector on the GameObjects. You may forgot attach the components.
Careless miss	• Forgotten to save the scripts • Target GameObject was disabled • Log message was disabled
When you ask it…	When you ask someone the problem, make sure you clarify the followings. • **What problems happened?** • **When the problems happened?** • **Were you able to identify where is problem?** • **How did you try to solve it so far?** • **What is your goal?**