



Construct 2

A game engine without the programming.



Construct 2 is a powerful tool

- Started as a prototyping program, but is now being used to make polished games.
- Has a GUI for level design and art.
- Uses programming logic without code.



Games made with Construct 2:

Prism Shell,
by Brooklyn Gamery

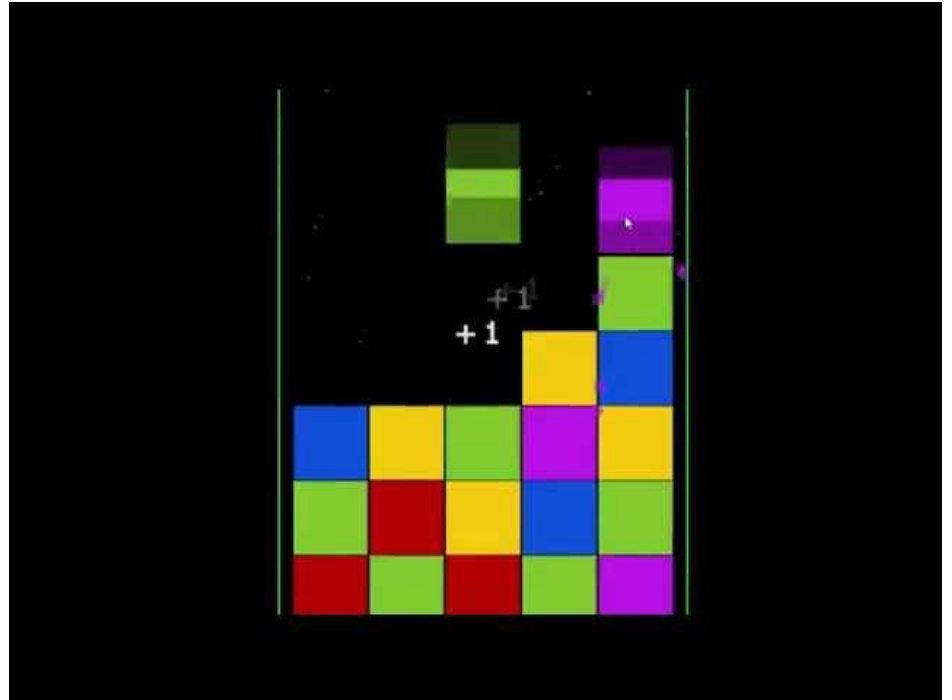




Games made with Construct 2:

Crush II,

By Arthur Ward Jr.





Games made with Construct 2:

The Next Penelope,
by Aurelien Regard





Today we will make a platforming game!

- Jump around platforms
- Collect rings (or some other awesome item)
- Don't touch enemies!

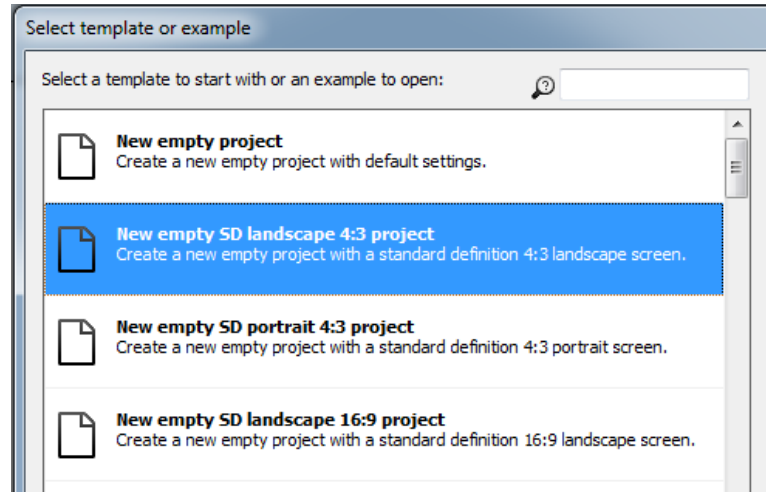


Basic concepts of C2



Projects

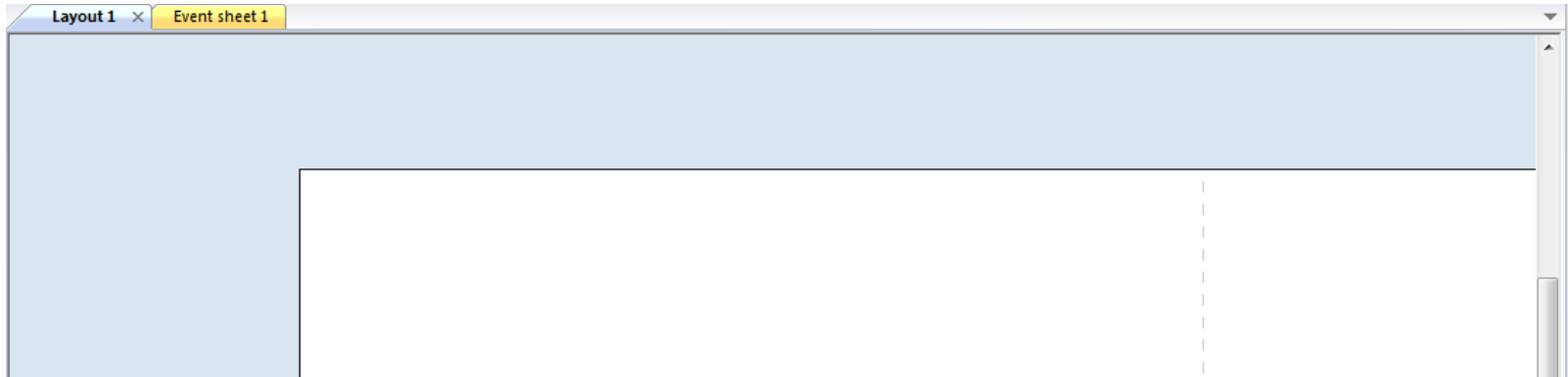
Construct comes with a bunch of premade project types. Make a new project.





Layouts

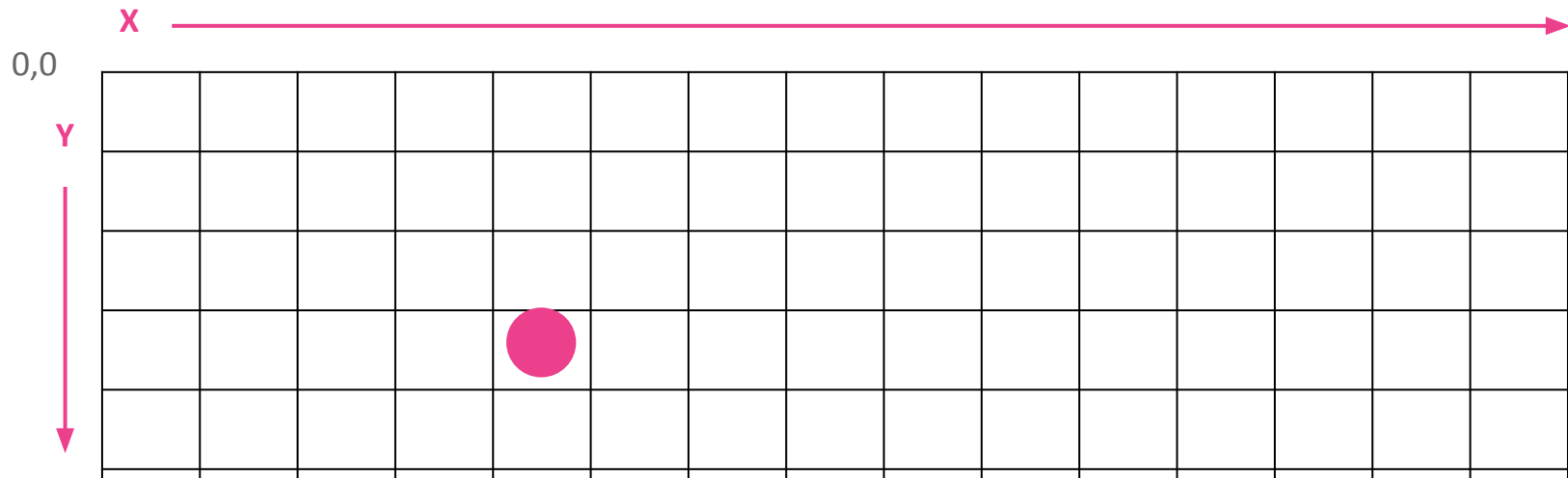
- Arrange characters, backgrounds, etc. on layers and move them around freely.
- Each object needs to be on a layout once.





Positioning

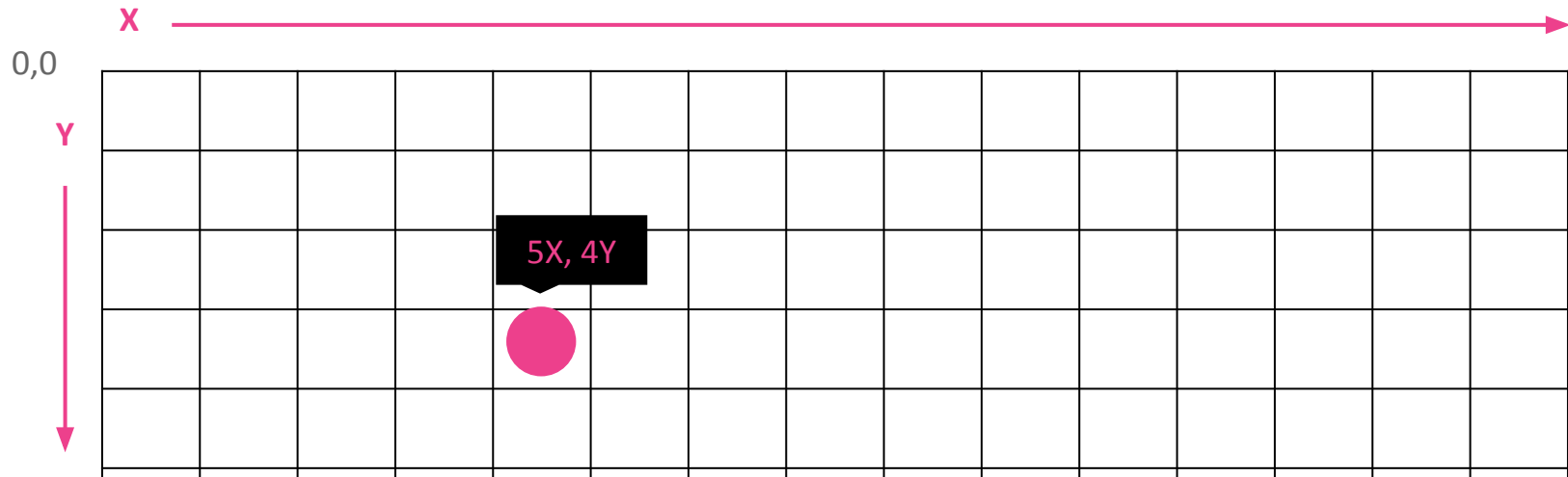
- X is horizontal
- Y is vertical





Positioning

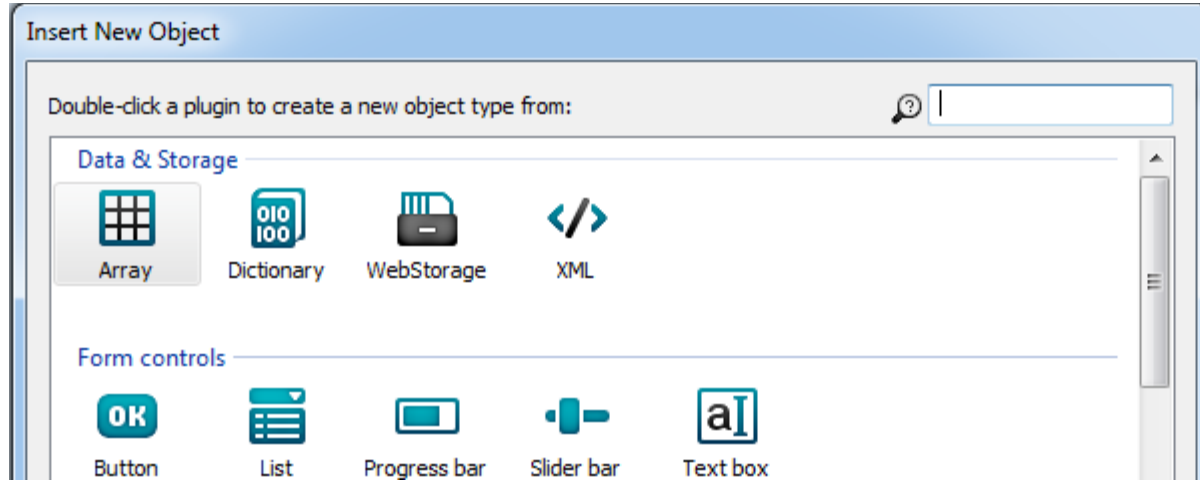
- X is horizontal
- Y is vertical





Objects

- You can create types of objects from plugins
- 3rd-party plugins can be downloaded & installed





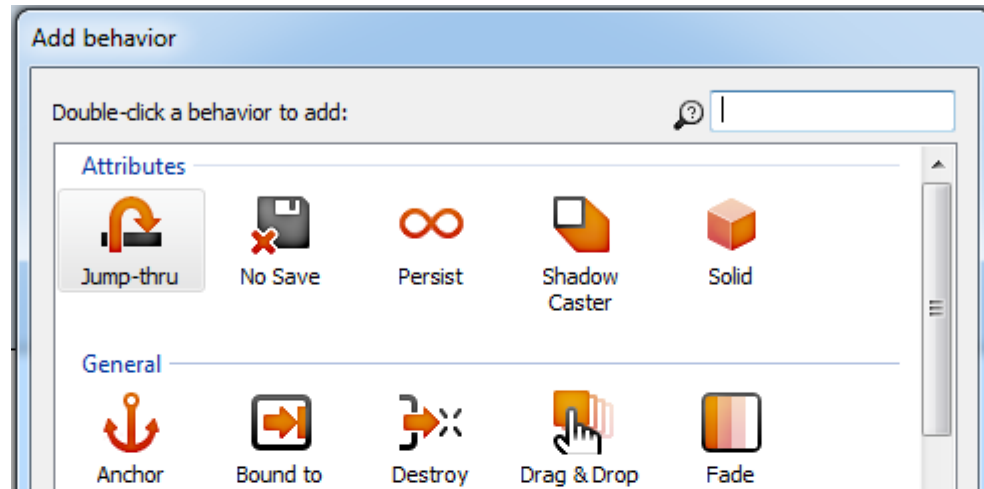
Challenge: make your game's objects

- Create a **sprite** that will be your player.
- Create a **sprite** to use for collectible items.
- Create a **sprite** to use for your enemies.
- Create a **9-patch** that will be used for platforms and walls.
- Arrange your objects on the layout.



Behaviors

Behaviors define what objects can do.





Challenge: give your objects behaviors

- Give your player **Platform** & **ScrollTo** behaviors.
- Give the walls and platforms the **Solid** behavior.

Instance variables

Add / edit [Instance variables](#)

Behaviors

Add / edit [Behaviors](#)

Effects

Blend mode	Normal
Add / edit	Effects






Container

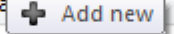
No container [Create](#)

Properties

Image	Edit
Left margin	16
Right margin	16


9patch: Behaviors






 Add new

Name	Type
------	------






Add behavior

Double-click a behavior to add: 

Attributes

 Jump-thru	 No Save	 Persist	 Shadow Caster	 S
---	---	---	---	---

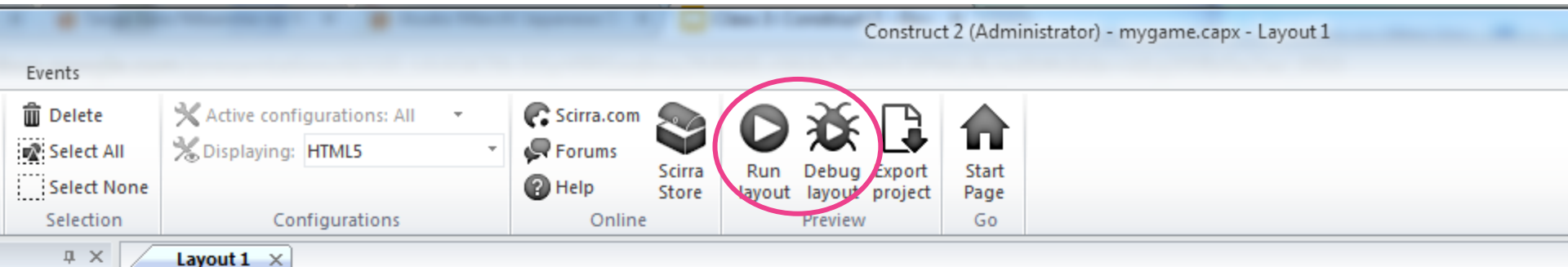
General

 Anchor	 Bound to layout	 Destroy outside layout	 Drag & Drop	 F
---	--	---	--	--



Now try running your game!

Try using your keyboard to move.





Event Sheets

Set up all kinds of actions and systems.

The screenshot shows the Construct 2 Event Sheet editor. The top toolbar contains tabs for different layouts: Layout 1, Event sheet 1, 2_Platform_Entrance, 1_Subway_Entrance, 00_Title Screen, 01_Scene1, and 000_Main_Controls (which is currently selected). Below the toolbar, the event sheet is displayed with several global text and number actions:

- Global text **CurrentLayout** = ""
- Global number **PlayerLocationX** = 135
- Global number **PlayerLocationY** = 1080

Below these, a system named "Encounter system" is expanded. The system configuration table is as follows:

System	Event	Action
System	On start of layout	Set CurrentLayout to <i>LayoutName</i>
SceneFa...		Move to layer 2
SceneFa...		Set width to <i>LayoutWidth</i>
SceneFa...		Set height to <i>LayoutHeight</i>
SceneFa...		Set position to <i>(0, 0)</i>
SceneFa...		Fade: start fade



Events Require Conditions

If a certain condition is true, something will happen.

In code:

```
if (x = 1) {  
    console.log("hello!");  
}
```



Question!

How do we get the player to look like it's moving in a certain direction?

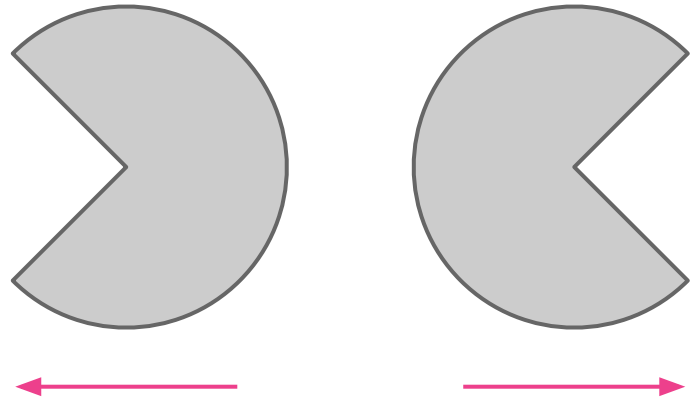


Question!

How do we get the player to look like it's moving in a certain direction?

Two different ways:

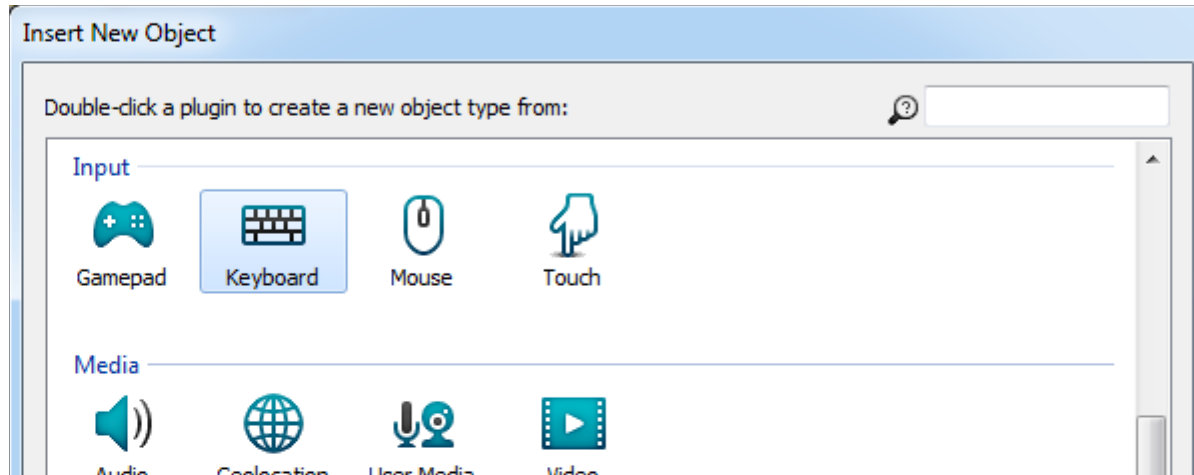
1. Mirror the object
2. Change animations





Keyboard Input

Before we can use keyboard events, add the keyboard plugin as an object.





Challenge: Create your first events

Create an event for each set of pseudo-code:

when the `left arrow key` is pressed,
the player should look left.

when the `right arrow key` is pressed,
the player should look right.



Challenge: Create ghost movement

Create events for this pseudo-code:

```
each frame(tick),  
move enemies in the direction of the  
player's position.
```

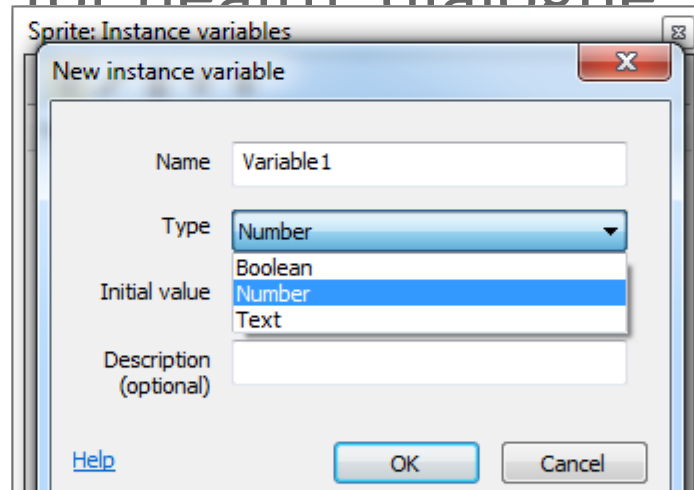


Scoring and Health



Variables

- Objects can have variables that store information
- Can be used for health, dialogue, score, etc.





Challenge: Set up variables

- Create a **number** variable for **coins**.
- Create a **number** variable for the player's **health** and set its initial value to **50**.

The screenshot shows the Construct 2 interface with three windows open:

- Instance variables**: A list of variables for the selected object. The 'Add / edit' button is circled in pink, and a pink arrow points from it to the 'Coin: Instance variables' window.
- Coin: Instance variables**: A window with a table for instance variables. The table has columns for Name, Type, and Initial value. A pink circle highlights the '+' button in the toolbar, with a pink arrow pointing to the 'New instance variable' window.
- Sprite: Instance variables**: A window titled 'New instance variable' with the following fields:
 - Name: Variable 1
 - Type: Number (selected in a dropdown menu)
 - Initial value: Number (selected in a dropdown menu)
 - Description (optional):



Challenge: Collision with coins

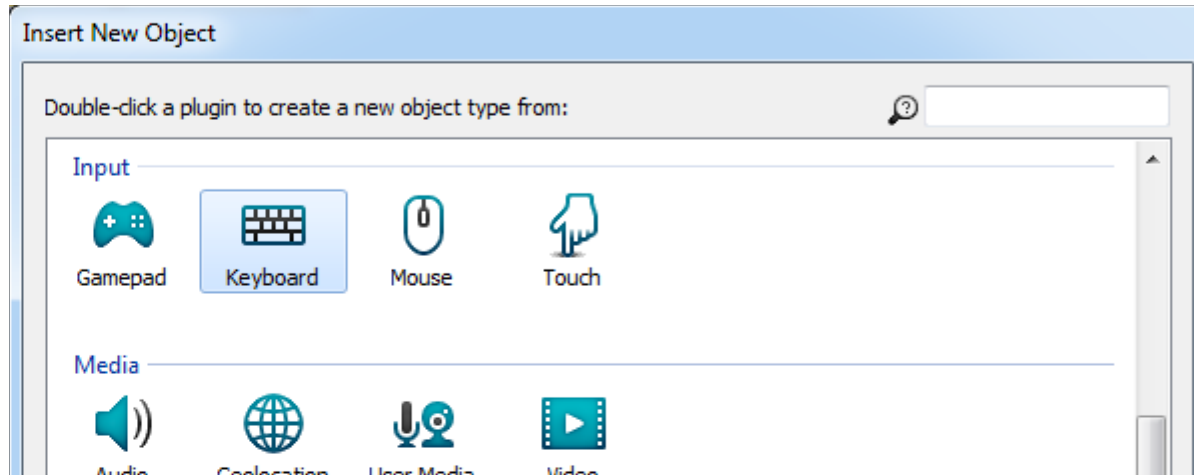
- Create an event for this pseudo-code:

```
on collision with coins,  
coin count should increase by 1.
```



Using Text

Text can be used for a variety of things, including the user interface (UI).





Challenge: Set up text

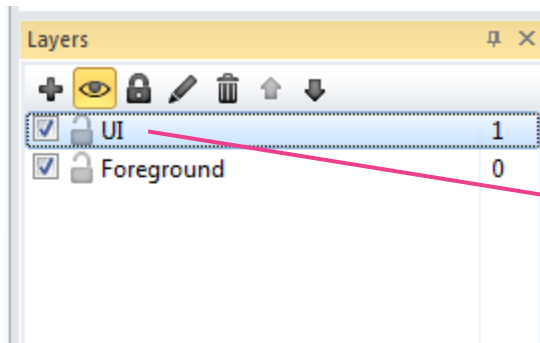
- Create a text object that will be used for **coins**.
- Create a text object that will be used for **health**.
- Give the text objects initial values
 - (I used “Coins: 0” and “Health: 50”).
- Place both objects on your layout and arrange them to your liking.



Setting up a UI layer

In order to get the text to stop moving out of view, create a **new layer** and set the **parallax** to **0,0**.

Don't forget to **move your UI onto the new layer!**



Layer properties	
Name	UI
Initial visibility	Visible
Background color	<input type="checkbox"/> 255, 255, 255
Transparent	Yes
Opacity	100
Force own texture	No
Scale rate	100
Parallax	0, 0
Editor properties	
Visible in editor	Yes



Referencing Variables

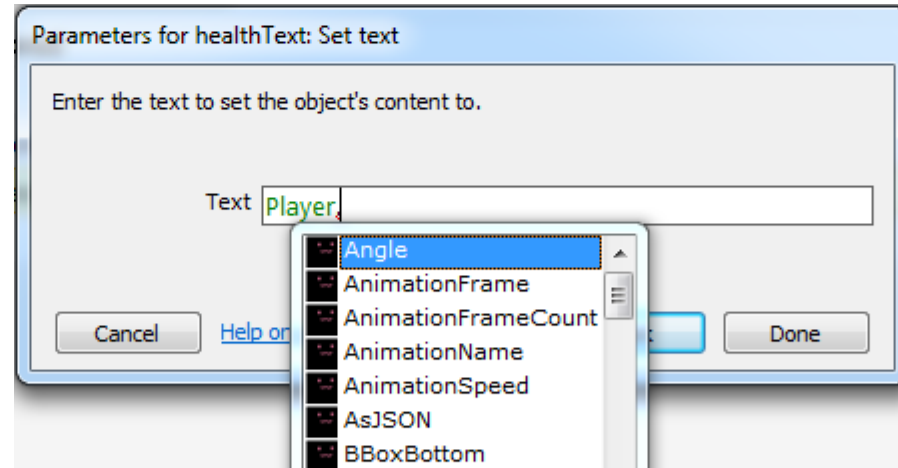
Access information about objects in addition to object variables using **dot notation**.

Examples:

```
Player.height
```

```
Player.width
```

```
Player.variableName
```





Combining strings and numbers

You can combine multiple types of data (strings, numbers, variables, etc.) by using the **&** symbol.

Examples:

```
"Layout width: " & LayoutWidth
```

```
"Position: " & Player.X & Player.Y
```

```
"My age is: " & 15
```




Doing math

You can do math using the following symbols:

- `+` (addition)
- `-` (subtraction)
- `/` (division)
- `*` (multiplication)



Challenge: Updating text

Create events for these sets of pseudo-code:

```
on collision with coins,  
set coin text to the number of coins.
```



Making enemies work



Challenge: make enemies move

Experiment with enemy movement using the **System's every tick** condition.

- Can you make enemies move toward the player?
- Away from the player?
- What other ways can you make enemies move?



Challenge: collision with enemies

Once you've found a movement style for your enemies, create events for this pseudo-code:

```
on collision with enemies,  
decrease player's health by 1,  
then set health text to player's health.
```



Challenge for the week!

If you can, work on your game some more.

- Find and import art assets.
- Make a background.
- Make a start and end screen.
- Link the gameplay layout to the start and end screens using **System** and **Keyboard events**.



Thanks! Questions?

@cattsmall
catt@codeliberation.org