Software Studio

軟體設計與實驗

Cocos Creator : Physics

Hung-Kuo Chu

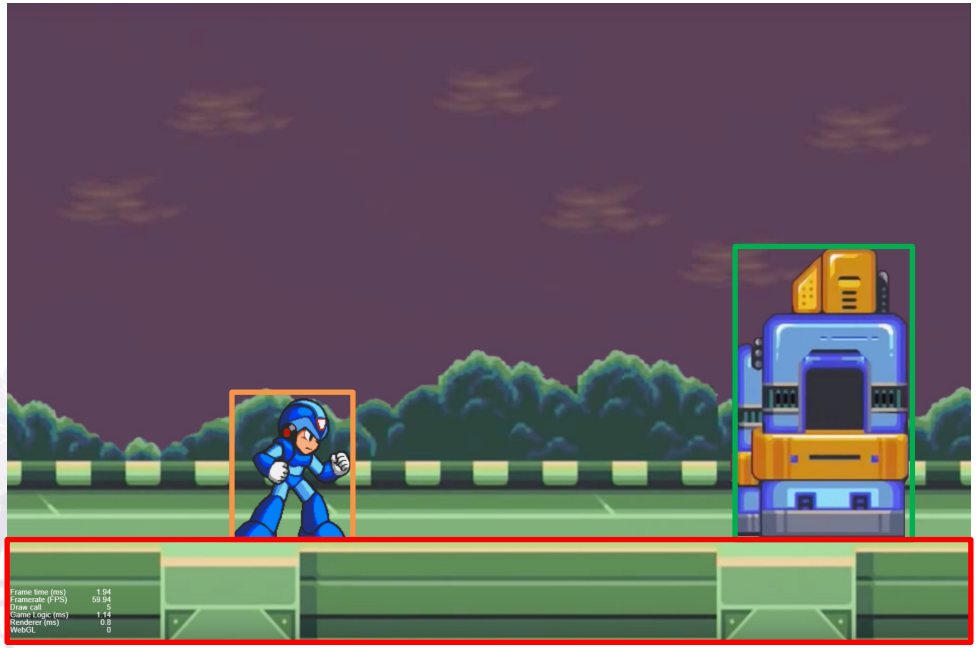
Department of Computer Science

National Tsing Hua University

CS2410

Physics System

- Simulate the real world
- Interaction between Nodes
- Collision, gravity, …, etc.



Physics Components

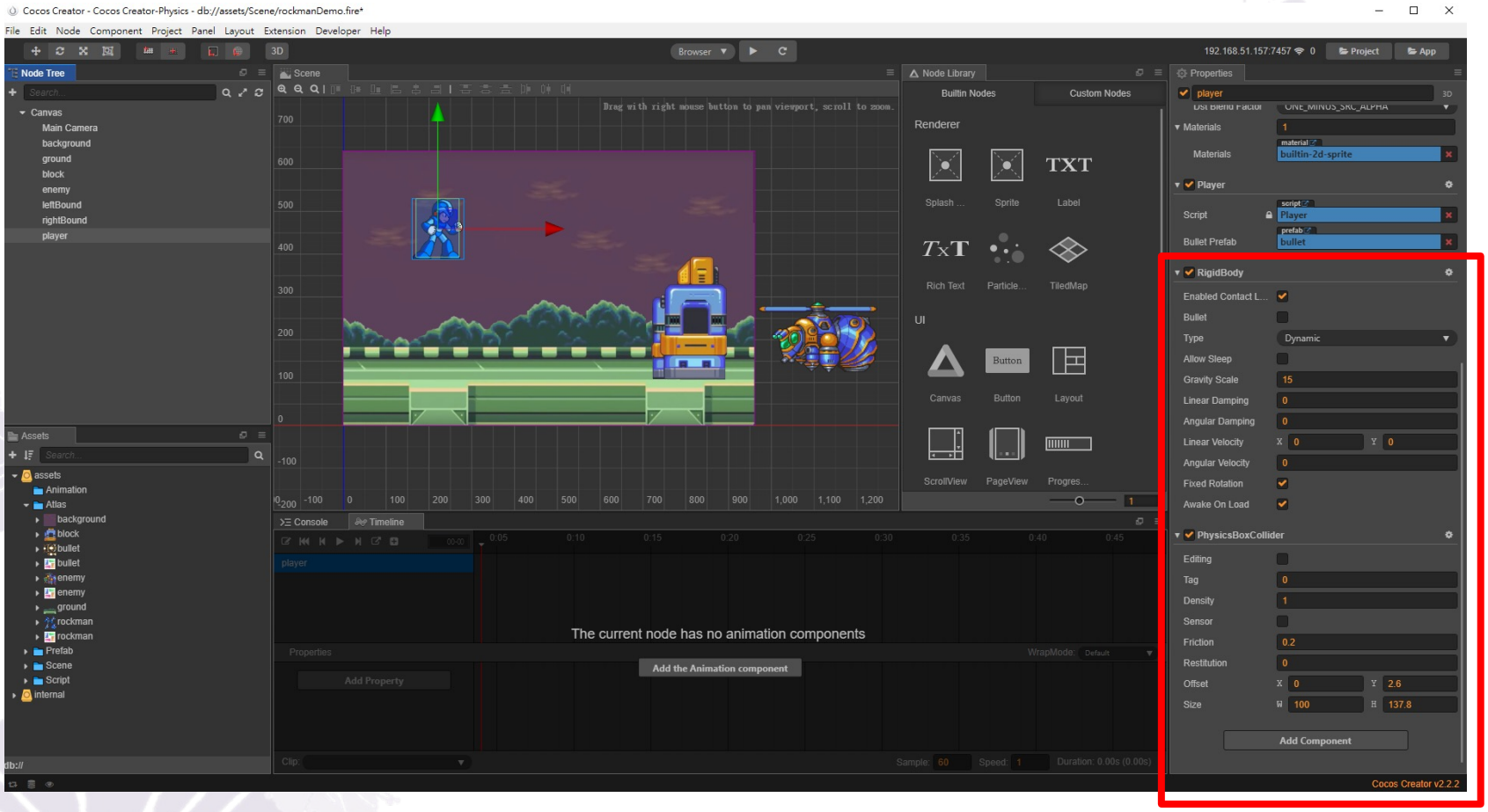
- **Rigid body**
  - The Node with rigid body component will become a dynamic object with mass, gravity…
- **Collider**
  - The shape of an object used for physical collisions.
- **Joint (Advanced)**
  - Simulate the interaction of real objects like ropes, wheels, …, etc.

Enable Physics Manager

```
onLoad () {

    cc.director.getPhysicsManager().enabled = true;
}
```

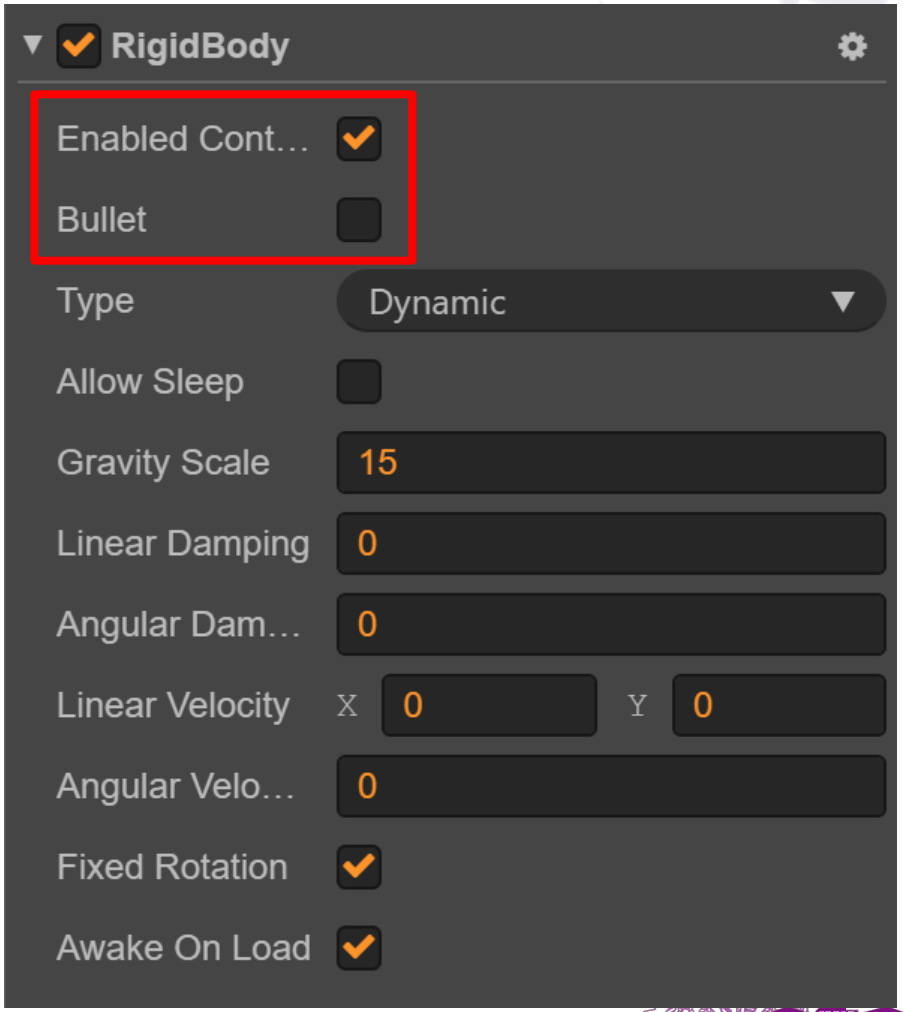
Any active script

Rigidbody & Collider



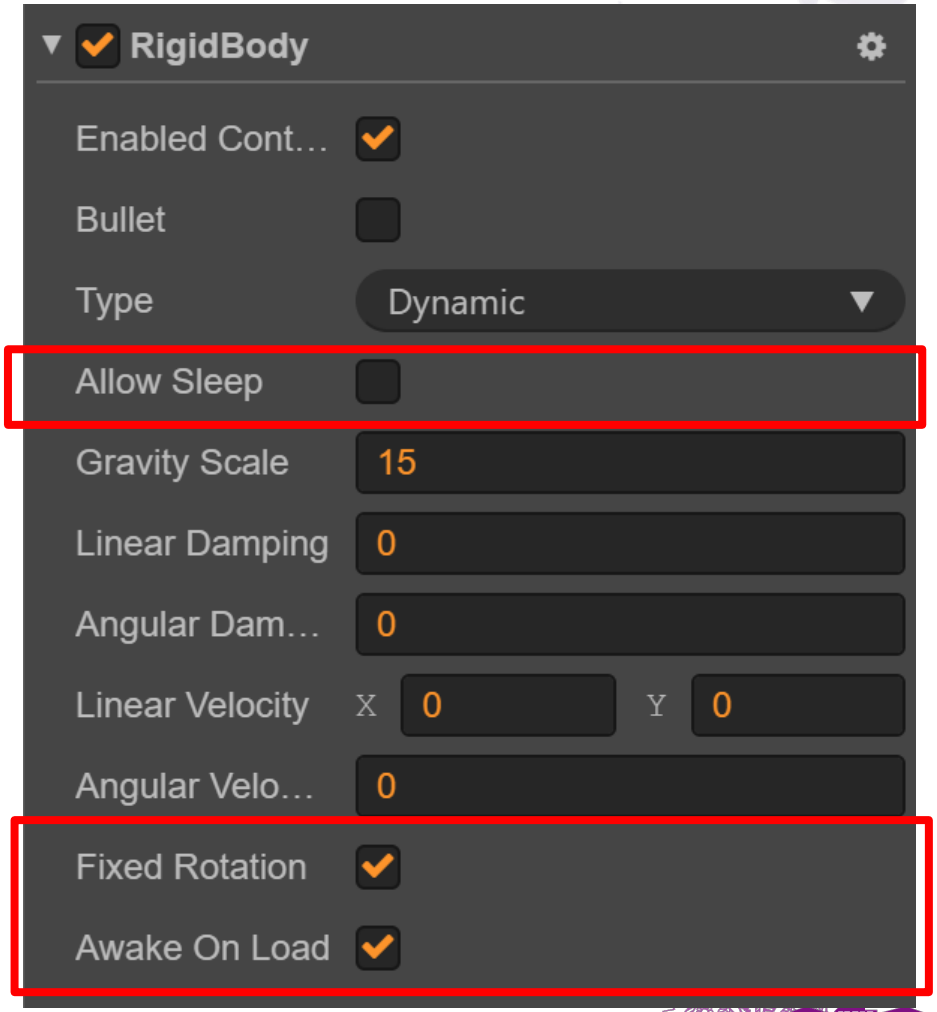
Rigid Body: Basic Properties

- **Enable contact listener:** send callback to the component of the node when collision happens
- **Bullet:** Prevent from tunneling through other moving bodies



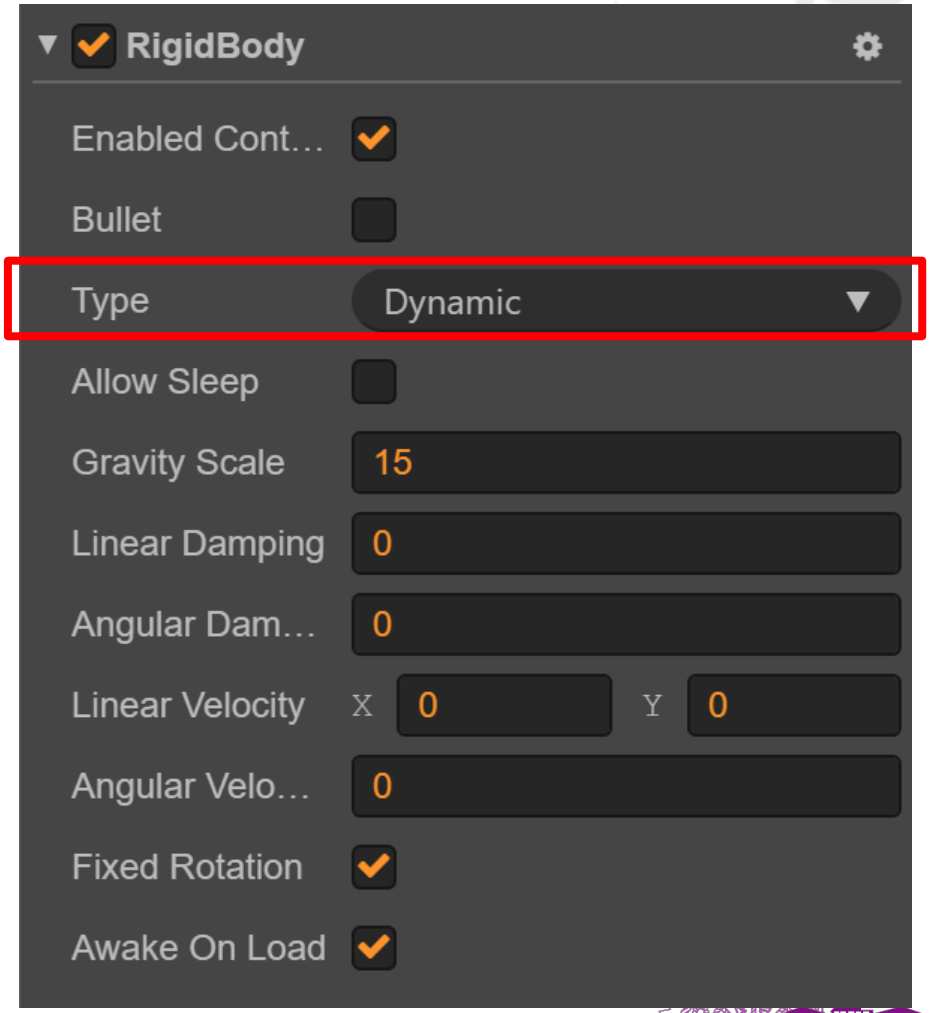
Rigid Body: Basic Properties

- **Allow Sleep:** whether the physics calculation of the rigidbody can fall asleep. It will increase CPU usage when user sets the flag to false.
- **Fixed Rotation:** we can prevent the node from rotating due to physics forces by setting the flag to true.
- **Awake On Load:** whether the node should have physics properties during initialization.



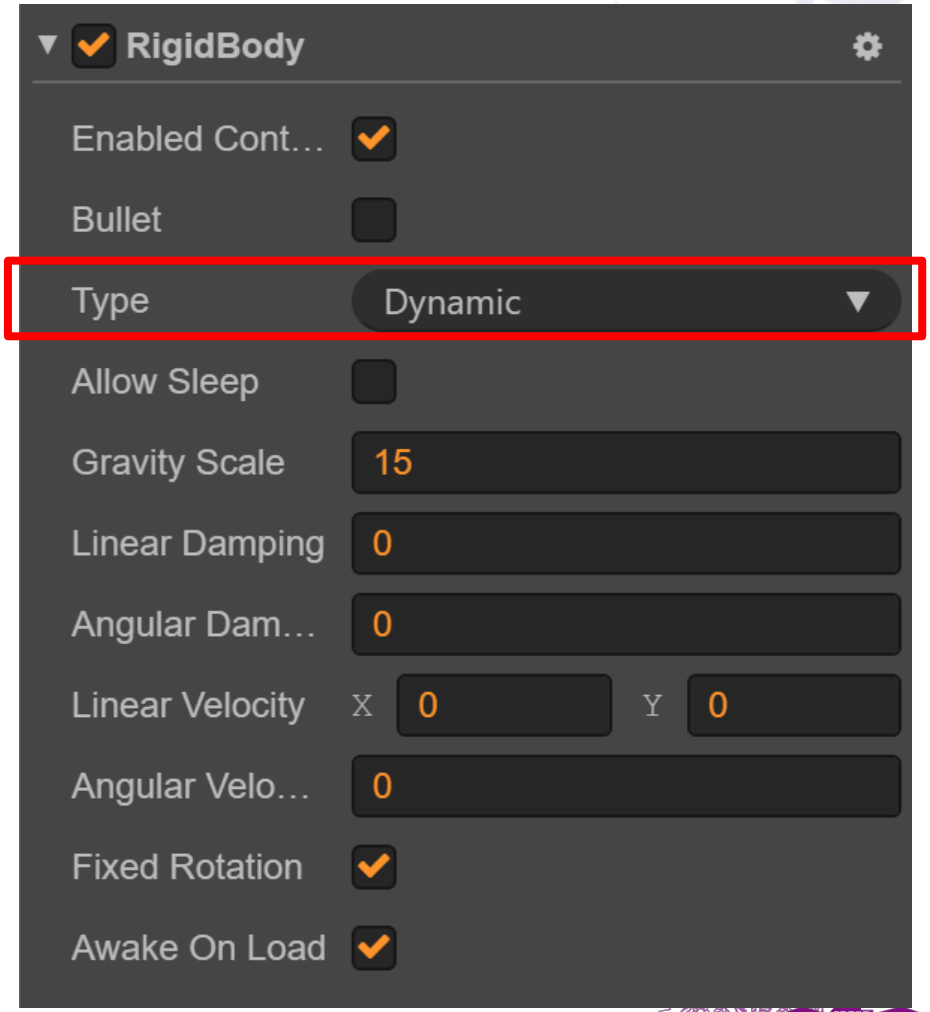
Rigid Body: Types

- **Static:** zero mass, zero velocity, that is not affected by gravity or force, but can set its position to move.
- **Dynamic:** with mass, its velocity can be set, will be affected by gravity.

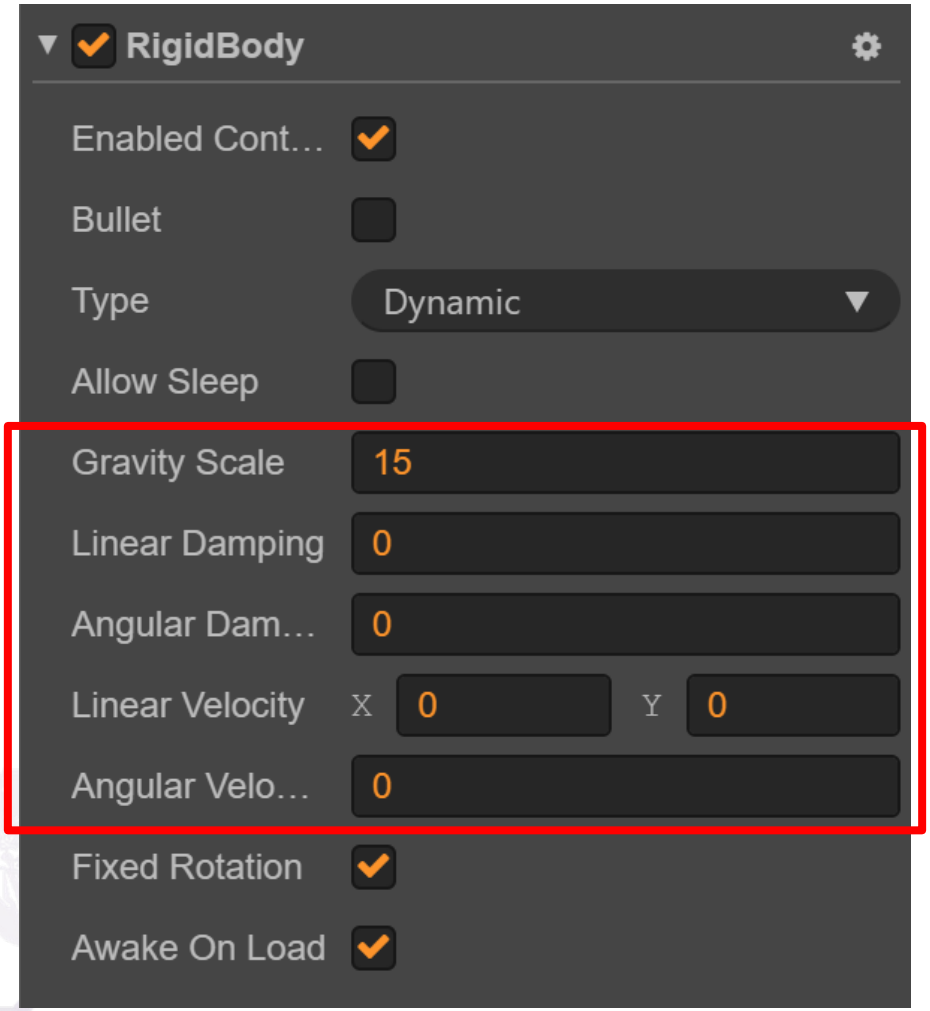


Rigid Body: Types

- **Kinematic:** zero mass, its velocity can be set, will not be affected by gravity, but can move by setting the velocity.
- **Animated:** derived from Kinematic type, mainly used for rigidbody and animation in combination (We won't cover this part).



Rigid Body: Physics Parameters



Collider

- Cocos Creator has two different types of collider system.
  - The colliders in **Collision System**
  - The colliders in **Physics System**

Collider in Collision System

- The collider in Collision System is used to detect whether two object collide each other.
- It doesn't produce any physics effects.
- We usually use it to calculate simple AABB collision detection or hit-test (for touch screen).
- We won't cover the collision system in this lecture.

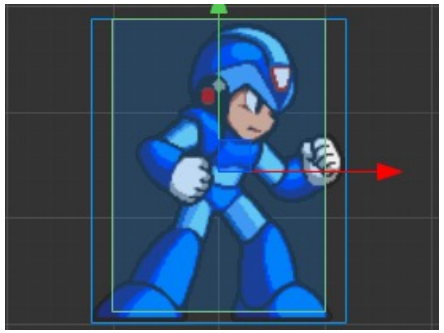
Ref: Collision Manager Scripting

Collider in Physics System

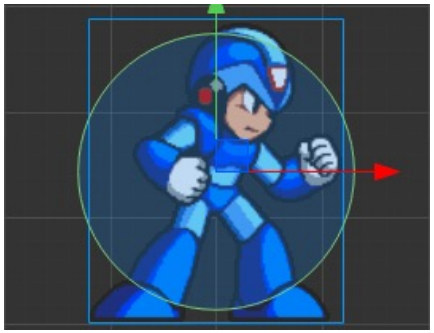
- The collider in Physics System inherits the collider from collision system.
- It deals with both **collision detection** and **physics effects**.
- We usually use it to simulate a physics world's collision, and use it to calculate such as raycast, force normal, etc.

Physics Collider: Types

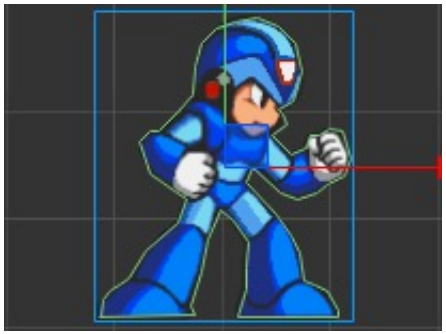
- Four types of physics collider



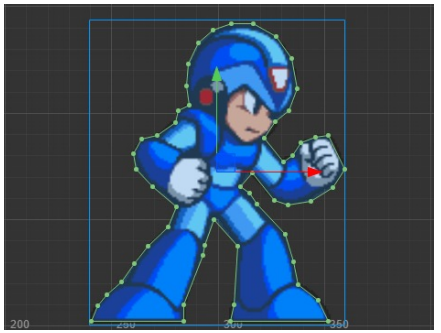
Box



Circle



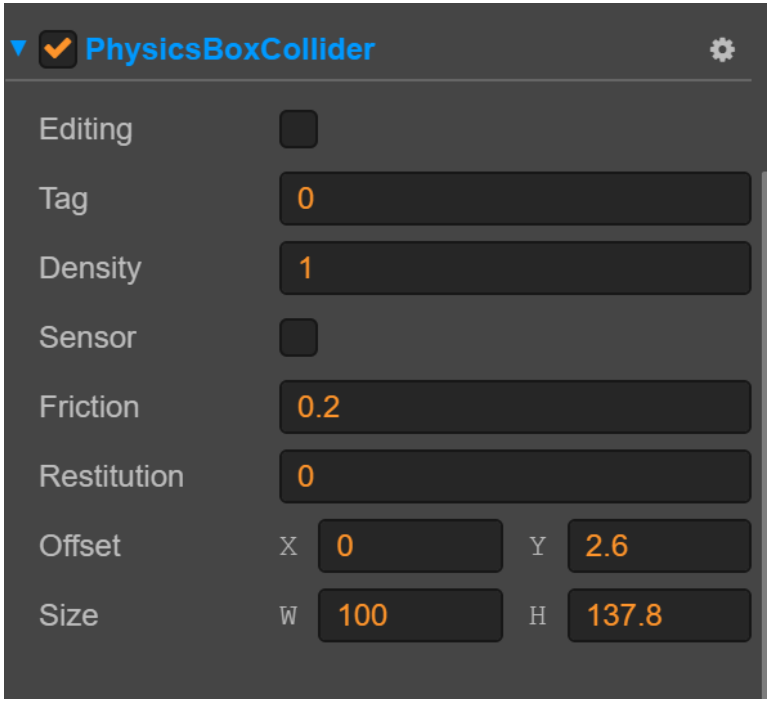
Chain



Polygon

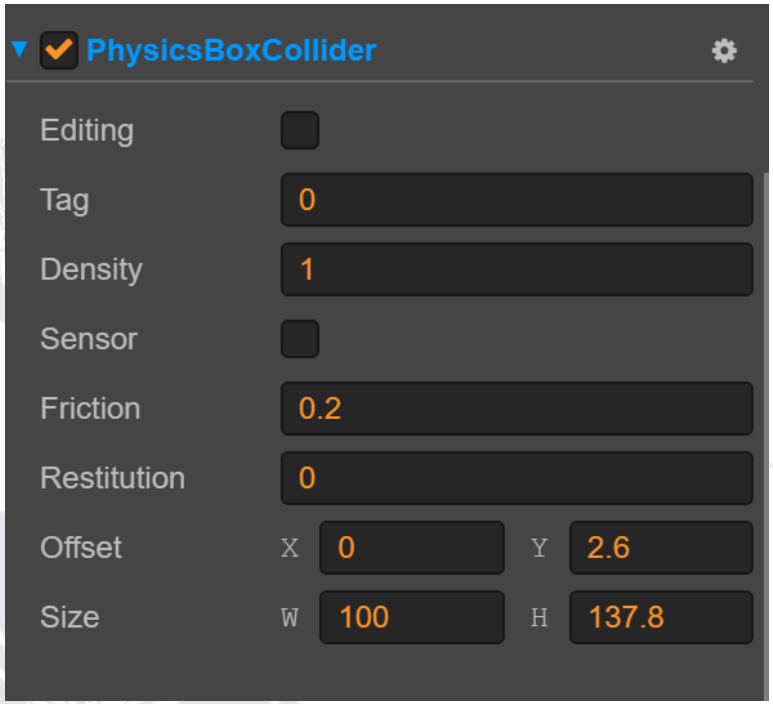
Physics Collider: Basic Properties

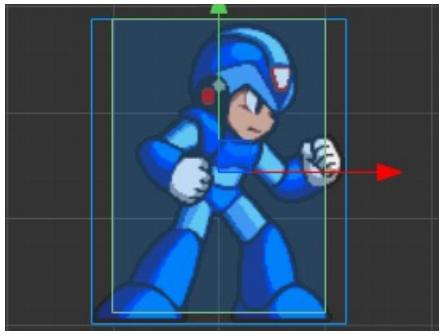
- **Editing:** edit collider shape freely
- **tag:** tag of the collider
- **Density:** density of the collider
- **Sensor:** like a trigger, no collision behavior occurs
- **Friction:** the friction of the collider
- **Restitution:** The elasticity of the collider



Box Collider

- **Offset:** position offset
- **Size:** size of the box

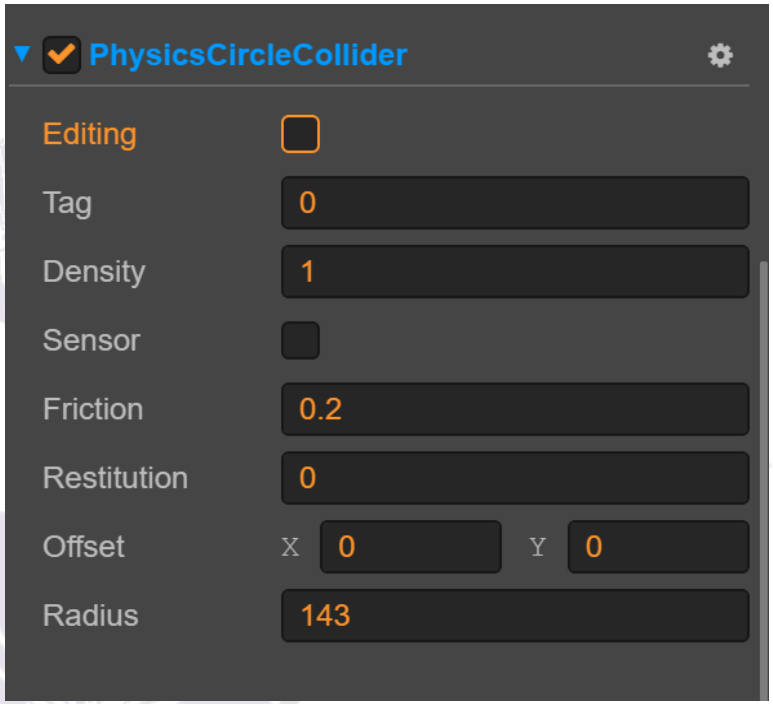


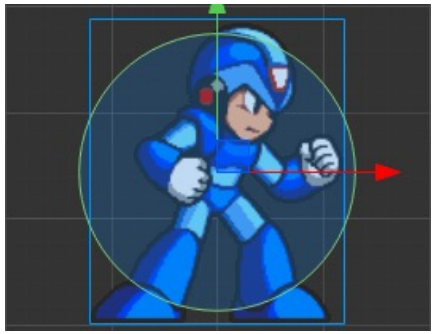


Box

Circle Collider

- **Offset:** position offset
- **Radius:** radius of the circle

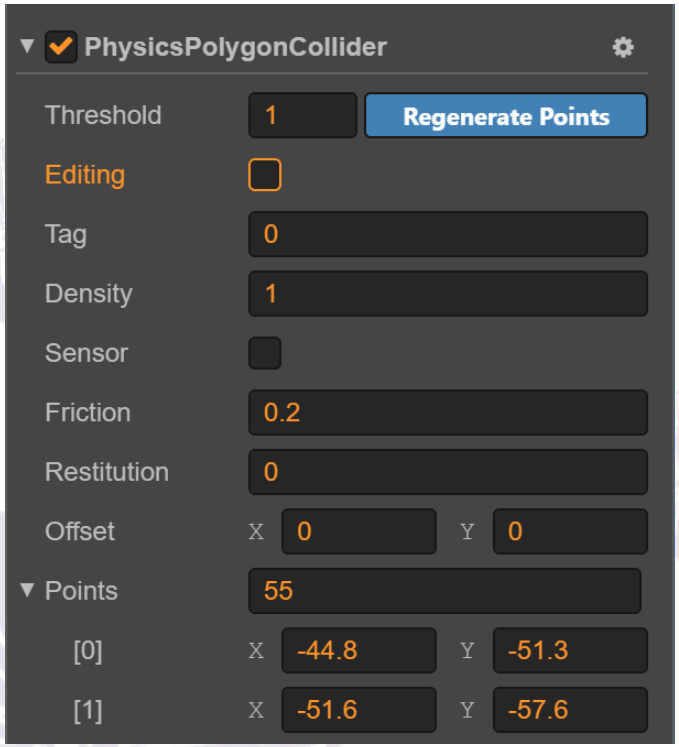


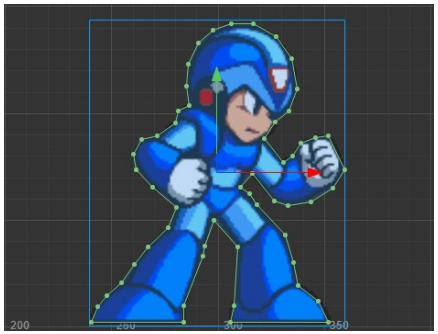


Circle

Polygon Collider

- **Offset:** position offset
- **Points[ ]:** position of boundary points

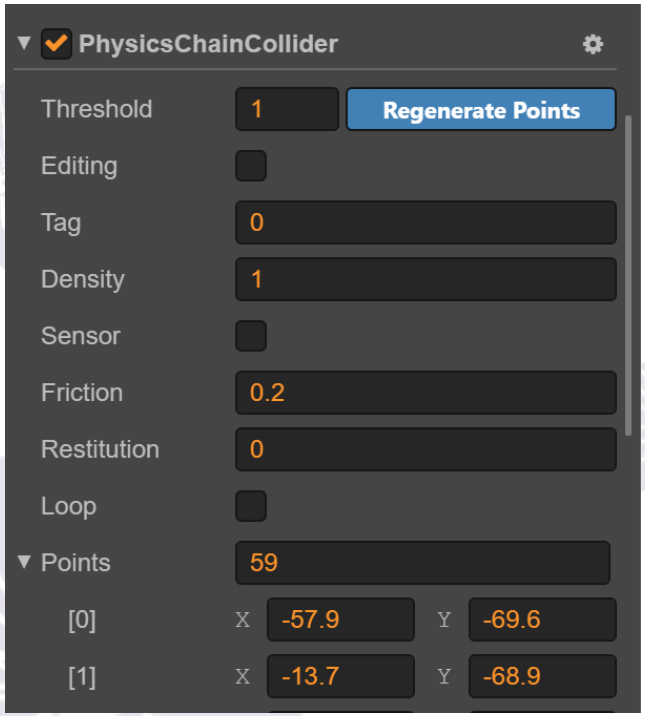


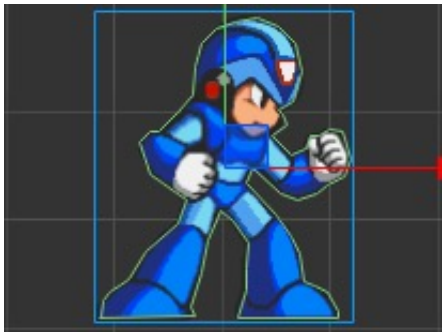


Polygon

Chain Collider

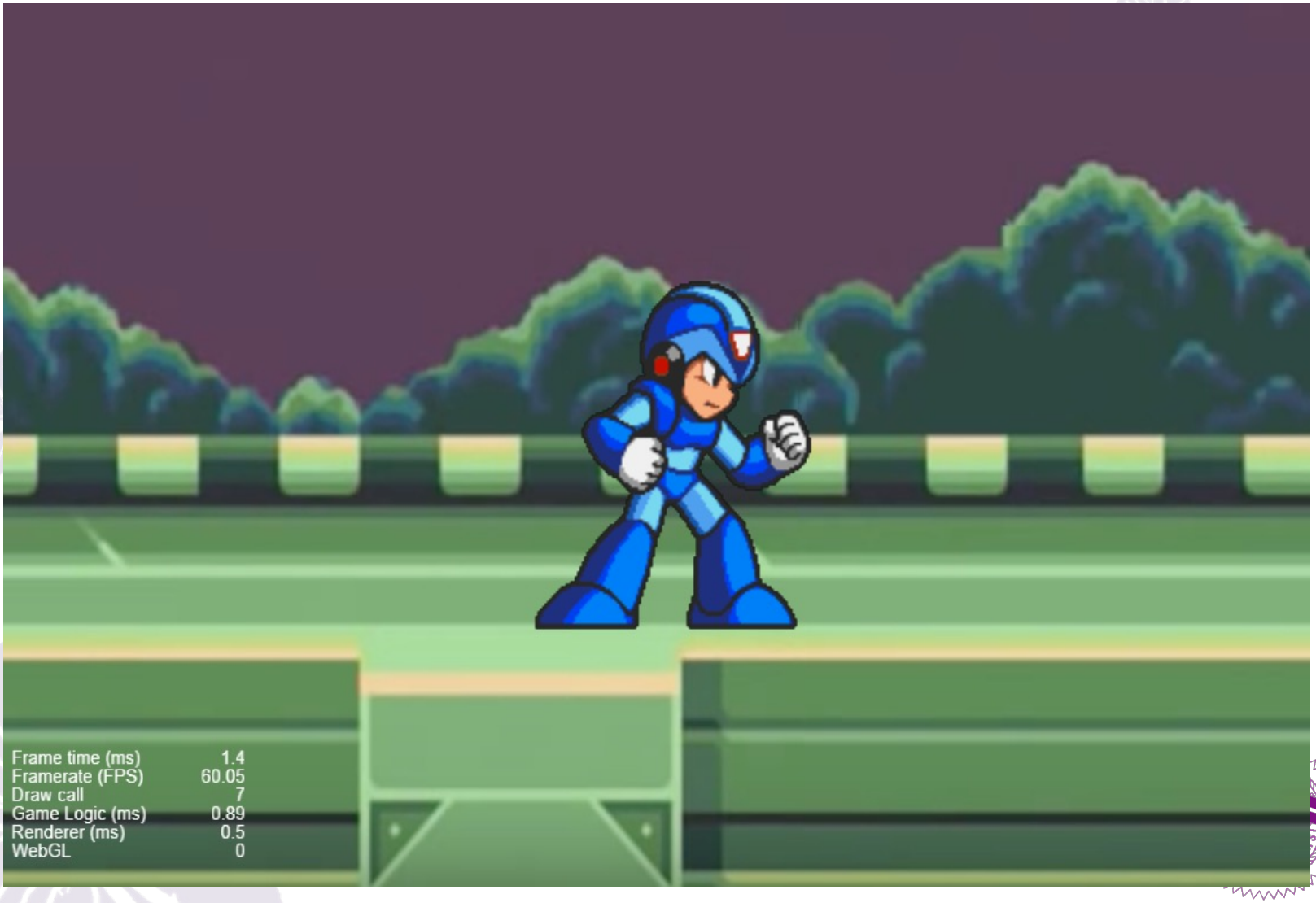
- **Loop:** Whether the chain forms a loop
- **Points[ ]:** position of boundary points, generated automatically



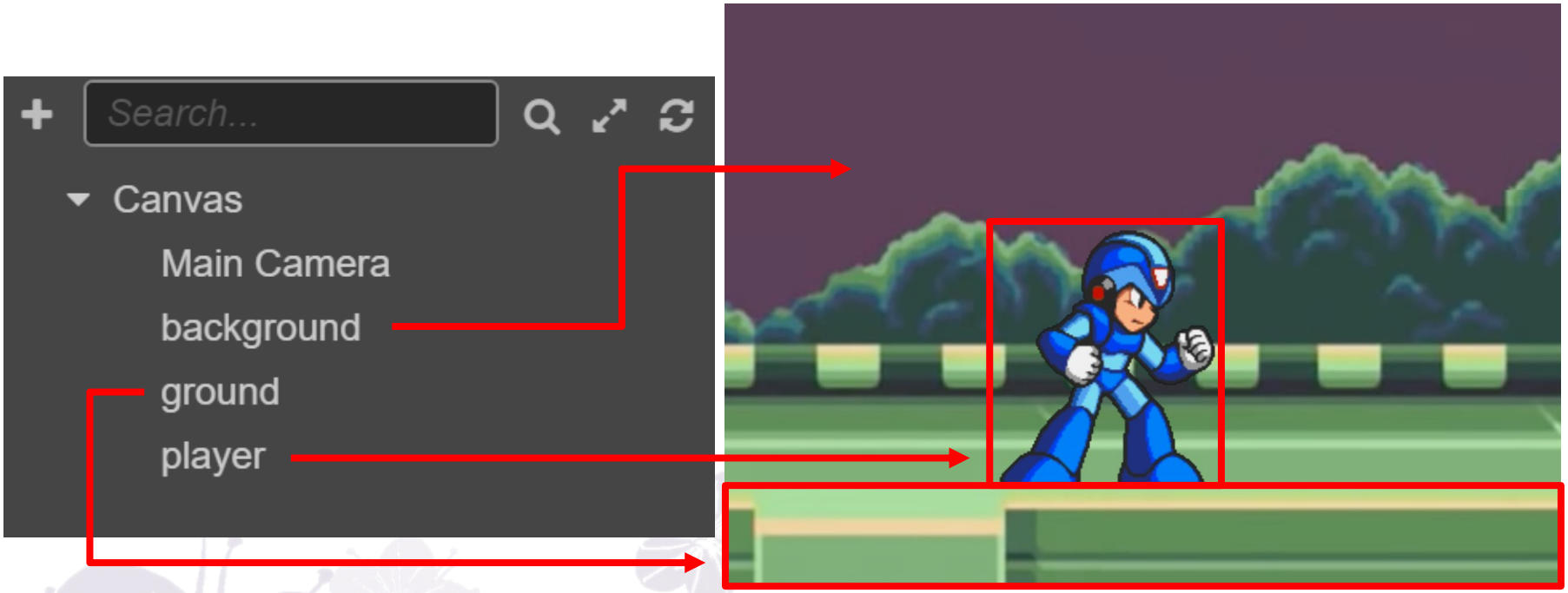


Chain

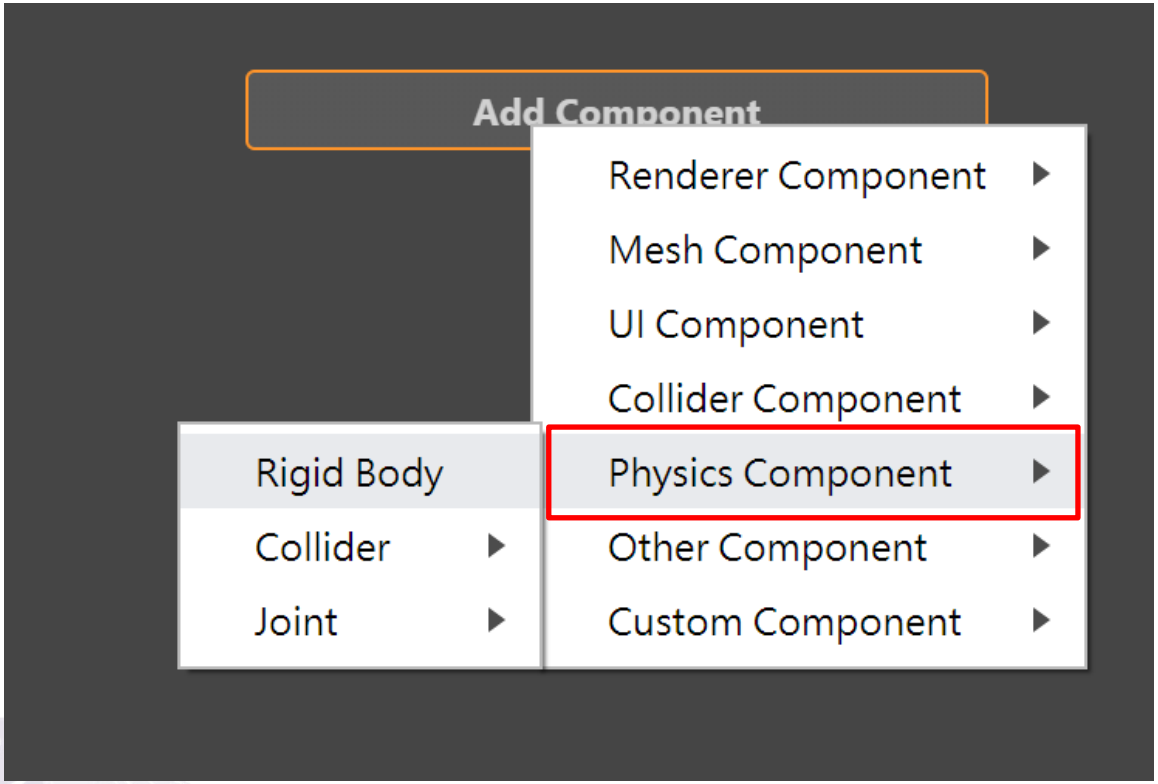
Make Rockman Stand on the Ground!



Make a Scene

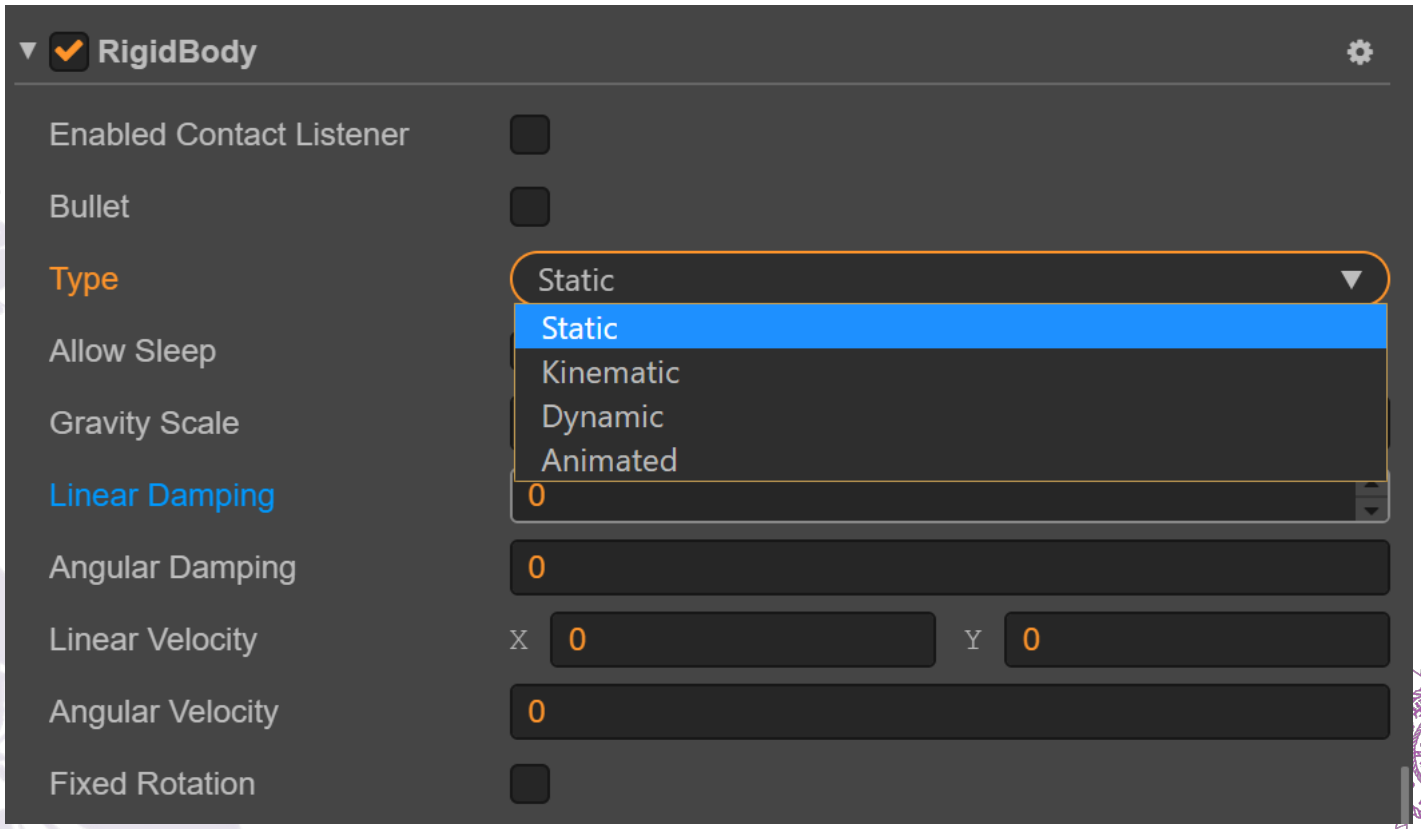


Add a RigidBody on the Ground

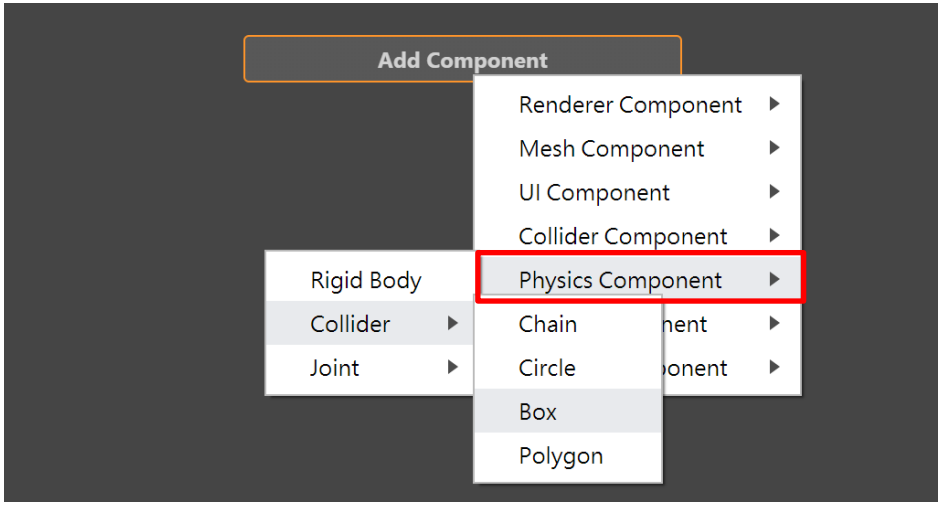


Using Static RigidBody

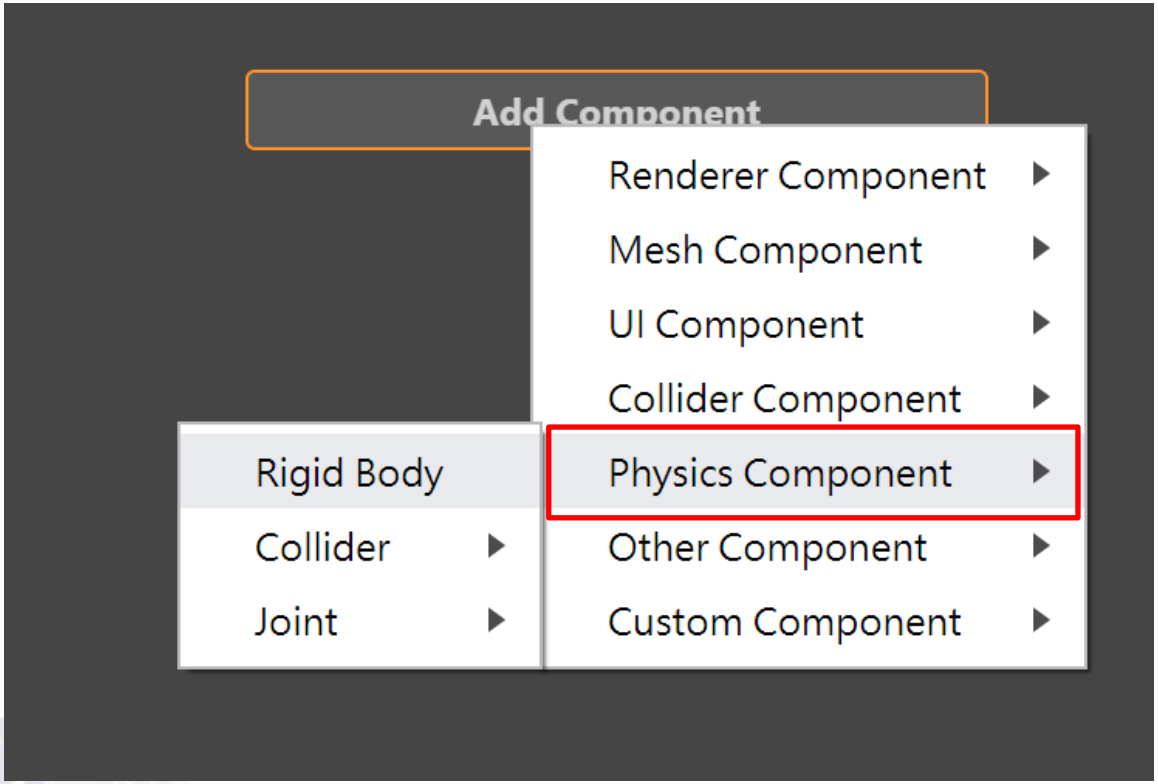
- Ground is not affected by gravity and force
  → **Static** type



Add a Collider on the Ground

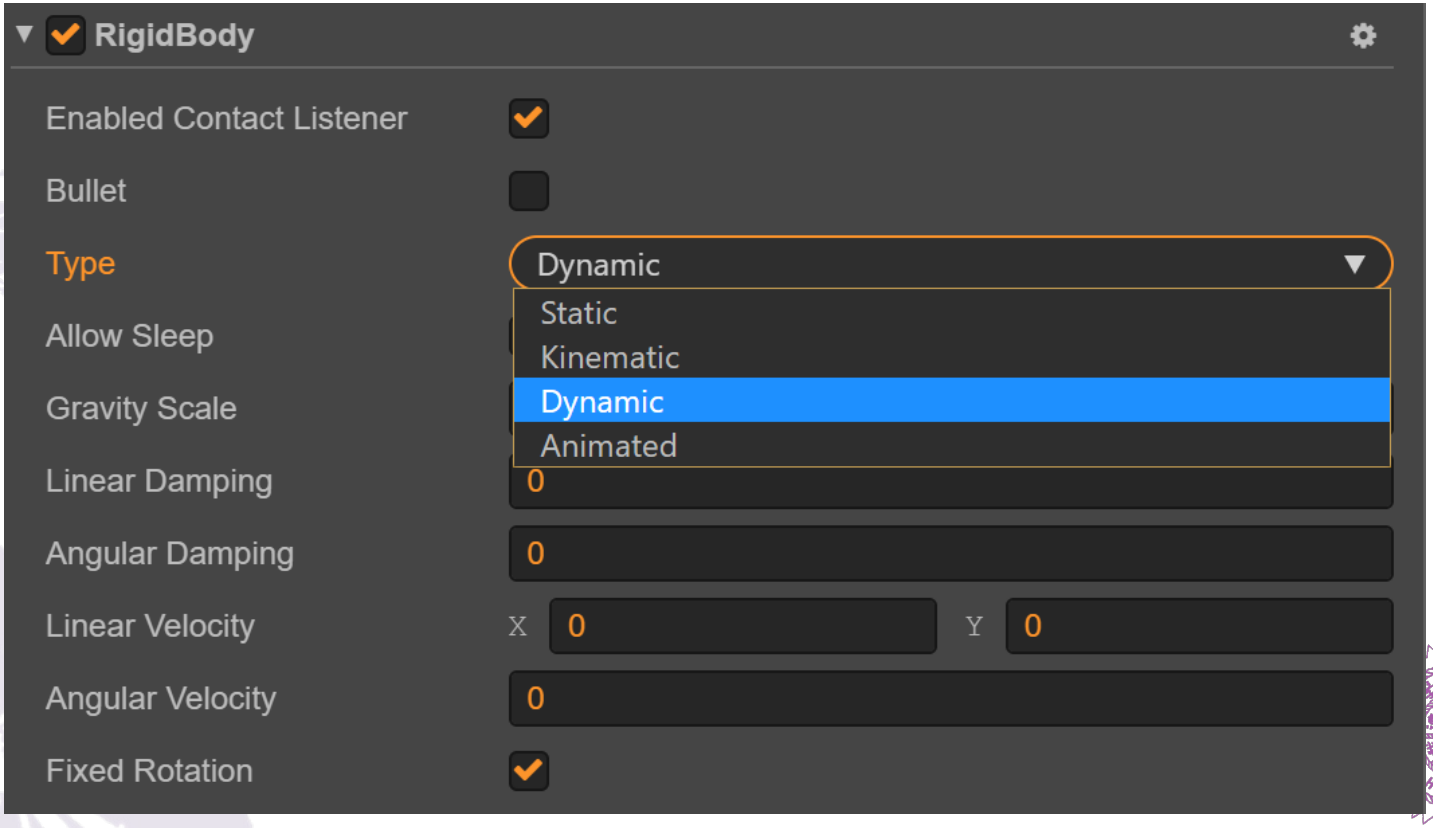


Add a RigidBody on Rockman

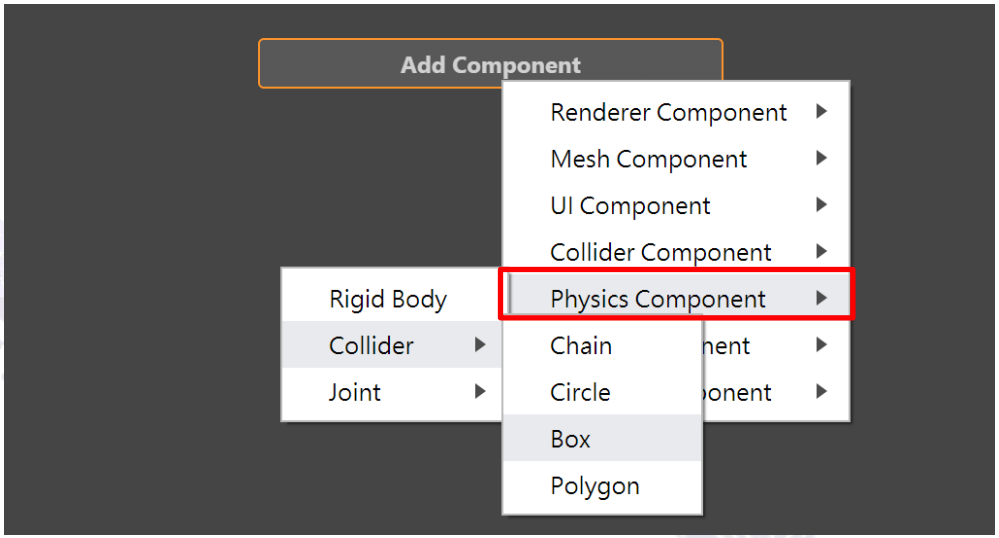


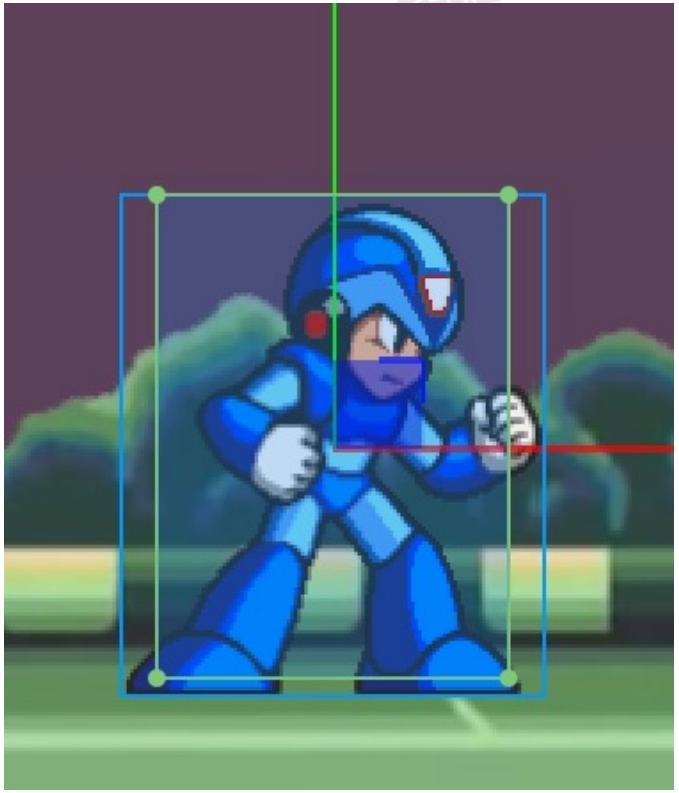
Using Dynamic RigidBody

- Rockman is affected by gravity and force
  → **Dynamic** type



Add a Collider on Rockman





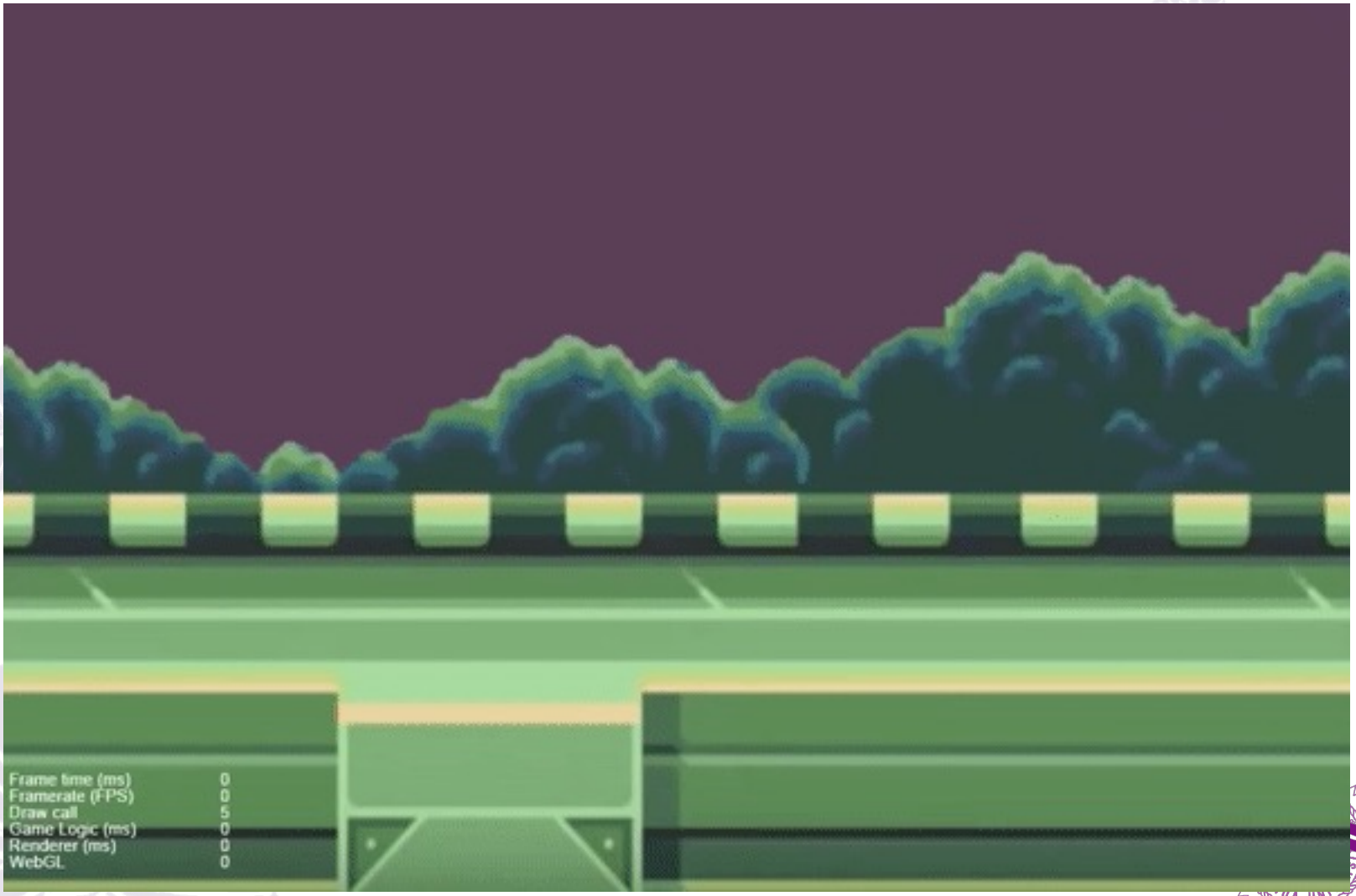
Enable Physics Manager

- Run the code in **Player.ts** to enable Physical System.

```
onLoad () {

    cc.director.getPhysicsManager().enabled = true;
}
```

Player.ts

Let’s Play!


Frame time (ms) 0
Framerate (FPS) 0
Draw call 5
Game Logic (ms) 0
Renderer (ms) 0
WebGL 0

Physics & Script

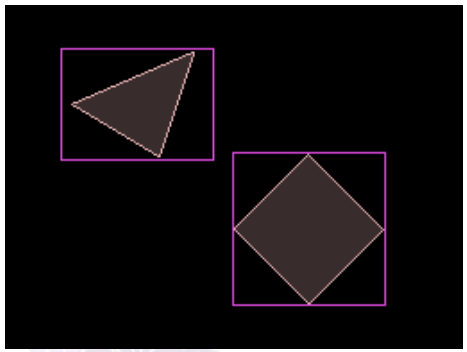
- What should Nodes do when collision happens?

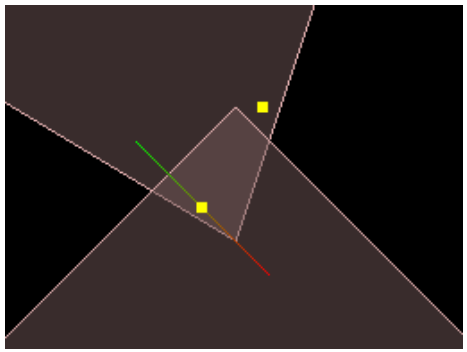
Collision Callback Functions

- **onBeginContact(contact, self, other)**
  - Callback when two colliders begin to contact.
- **onEndContact(contact, self, other)**
  - Callback when contact is just about to end.
- **onPreSolve(contact, self, other)**
  - Callback **before** the physics engine handle the collision behavior.
- **onPostSolve(contact, self, other)**
  - Callback **after** the physics engine handle the collision behavior.

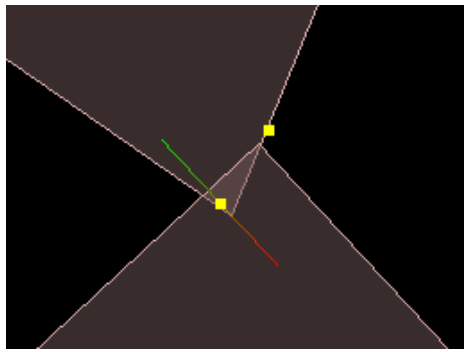
Ref: Contact Callback

Collision Callback Order

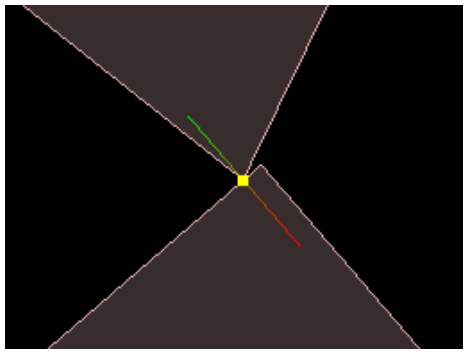




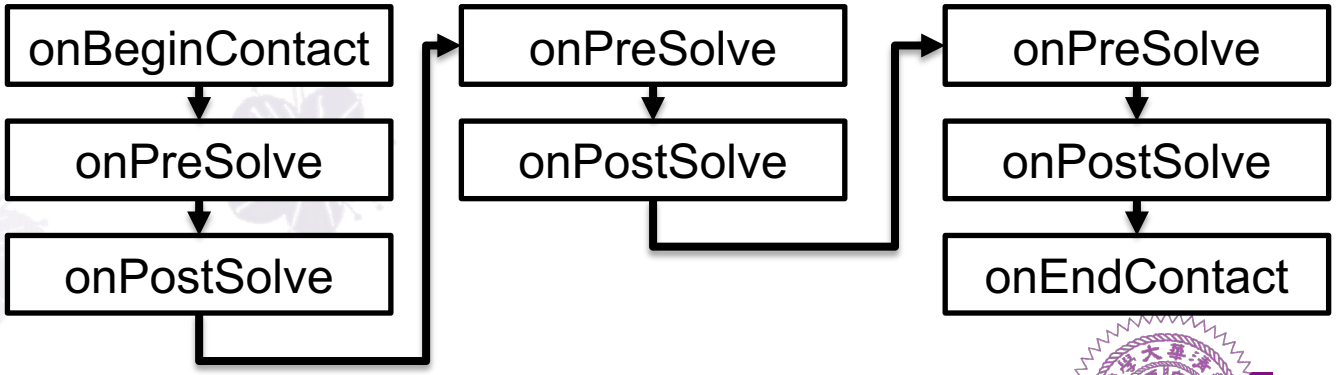
Collision 1



Collision 2



Collision 3



Collision Callback Parameters

- **contact**
  - The information about the collision
- **self**
  - The collider component of the Node
- **other**
  - The collider component of another Node

Ref: Contact Callback

Collision Detection

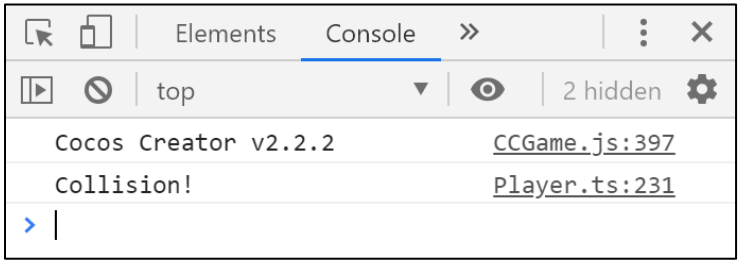
- Create a script on Player:

```
onLoad () {
        // enable physics system
        cc.director.getPhysicsManager().enabled = true;
}

onBeginContact(contact, self, other) {
        cc.log("Collision!");
}
```

Player.ts

Collision Detection

- Result

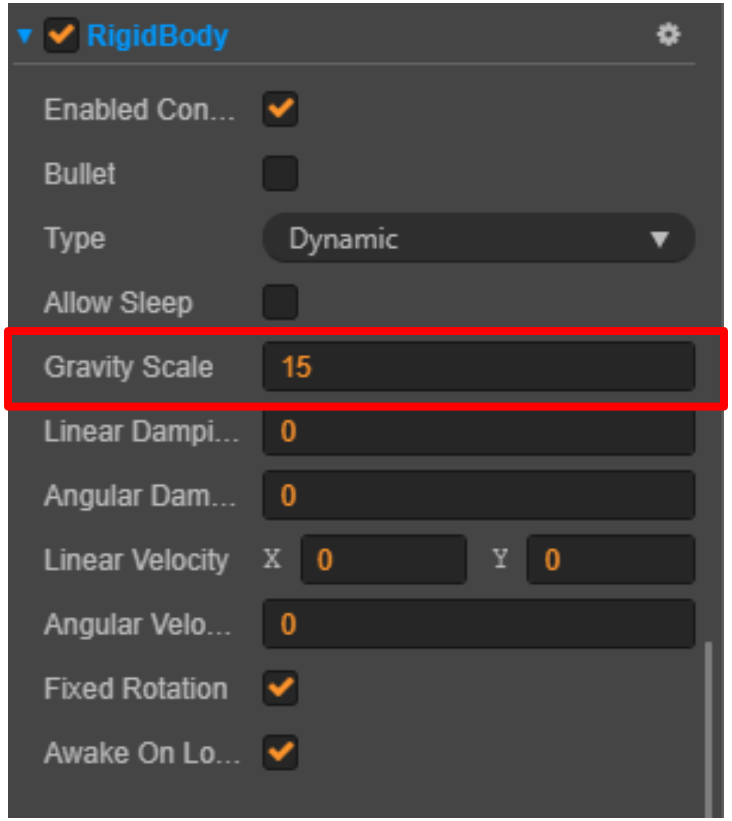


Let’s Jump !!!

- Adjust gravity scale of rigidbody on player
- Add keyboard events on the Player script
  - Method I: Apply **Force** to rigidbody
  - Method II: Change **velocity** of rigidbody

RigidBody Setting

- Set gravity scale to 15



Keyboard Events

```
onLoad() {
        cc.director.getPhysicsManager().enabled = true;
        // keyboard event
        cc.systemEvent.on(cc.SystemEvent.EventType.KEY_DOWN, this.onKeyDown, this);
        cc.systemEvent.on(cc.SystemEvent.EventType.KEY_UP, this.onKeyUp, this);
}

onKeyDown(event) {
        cc.log("Key Down: " + event.keyCode);
        if(event.keyCode == cc.macro.KEY.k) // press key k to jump
                this.kDown = true;
}

onKeyUp(event) {
        if(event.keyCode == cc.macro.KEY.k)
                this.kDown = false;
}

onBeginContact(contact, self, other) {
    cc.log("Rockman hits the ground");
    this.onGround = true;
}
```

Player.ts

Apply Force to RigidBody

```
private playerMovement(dt) {
   // The player jumps when the key is pressed, and it is standing on the ground
   if(this.kDown && this.onGround)
            this.jump();
}

private jump() {
    this.onGround = false;

    // add an up force directly to the mass of the rigid body
   this.getComponent(cc.RigidBody).applyForceToCenter(new cc.Vec2(0, 1500000));
}

update(dt) {
            this.playerMovement(dt);
}
```

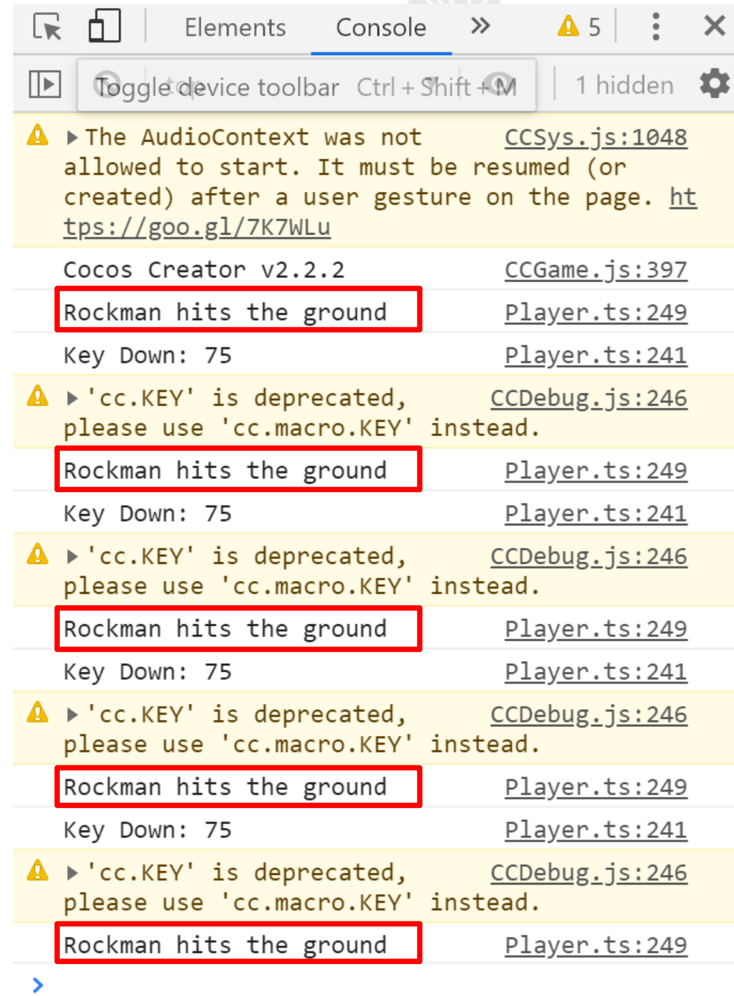
Player.ts

Change Velocity of RigidBody

```
private playerMovement(dt) {
    if(this.kDown && this.onGround)
                        this.jump();
}

private jump() {
    this.onGround = false;
    this.getComponent(cc.RigidBody).linearVelocity = cc.v2(0, 1500);
}

update(dt) {
    this.playerMovement(dt);
}
```

Player.ts

Let’s Play!


Elements
Console
5
Toggle device toolbar Ctrl + Shift + M
1 hidden
The AudioContext was not allowed to start. It must be resumed (or created) after a user gesture on the page. https://goo.gl/7K7WLu
CCSys.js:1048
Cocos Creator v2.2.2
CCGame.js:397
Rockman hits the ground
Player.ts:249
Key Down: 75
Player.ts:241
'cc.KEY' is deprecated, please use 'cc.macro.KEY' instead.
CCDebug.js:246
Rockman hits the ground
Player.ts:249
Key Down: 75
Player.ts:241
'cc.KEY' is deprecated, please use 'cc.macro.KEY' instead.
CCDebug.js:246
Rockman hits the ground
Player.ts:249
Key Down: 75
Player.ts:241
'cc.KEY' is deprecated, please use 'cc.macro.KEY' instead.
CCDebug.js:246
Rockman hits the ground
Player.ts:249
Key Down: 75
Player.ts:241
'cc.KEY' is deprecated, please use 'cc.macro.KEY' instead.
CCDebug.js:246
Rockman hits the ground
Player.ts:249

Let’s Move !!!

- Add keyboard events on the Player script
  - Change the **x** position of the player’s node

Keyboard Events: Revised

```
onKeyDown(event) {
        cc.log("Key Down: " + event.keyCode);
        if(event.keyCode == cc.macro.KEY.z) { // press key z to turn left
                this.zDown = true;
                this.xDown = false;
        } else if(event.keyCode == cc.macro.KEY.x) { // press key x to turn right
                this.xDown = true;
                this.zDown = false;
        } else if(event.keyCode == cc.macro.KEY.k) { // press key k to jump
                this.kDown = true;
        }
}

onKeyUp(event) {
        if(event.keyCode == cc.macro.KEY.z)
                this.zDown = false;
        else if(event.keyCode == cc.macro.KEY.x)
                this.xDown = false;
        else if(event.keyCode == cc.macro.KEY.k)
                this.kDown = false;
}
```

Player.ts

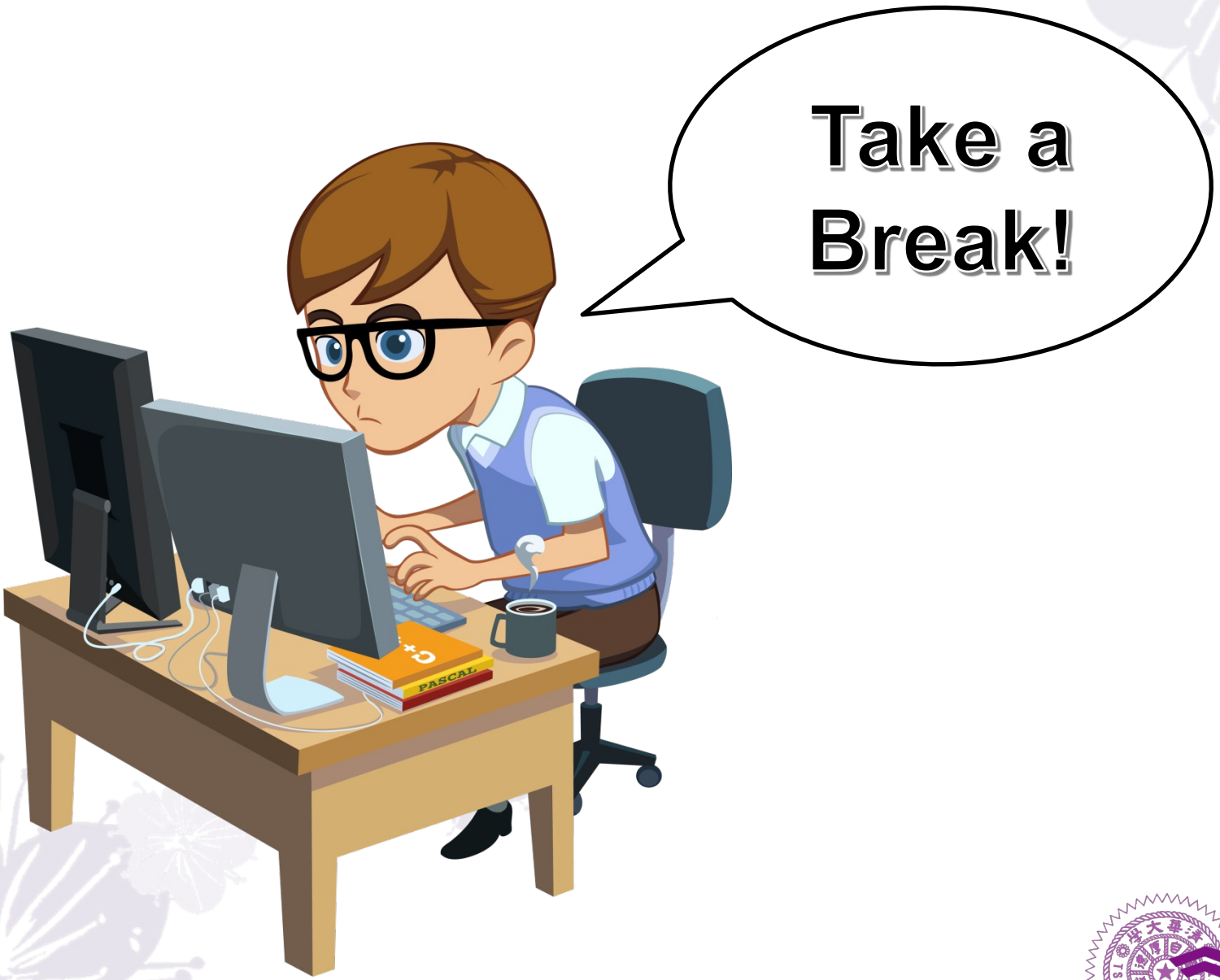
playerMovement()

```
private playerMovement(dt) {
    this.playerSpeed = 0;
    if(this.zDown) {
        this.playerSpeed = -300;
        this.node.scaleX = -1;  // modify node's X scale value to change facing direction
    } else if(this.xDown) {
        this.playerSpeed = 300;
        this.node.scaleX = 1;  // modify node's X scale value to change facing direction
    }
    this.node.x += this.playerSpeed * dt; // moving the object

    if(this.kDown && this.onGround) this.jump(); // handle the jump action

}

update(dt) {
            this.playerMovement(dt);
}

onBeginContact(contact, self, other) { … }
```

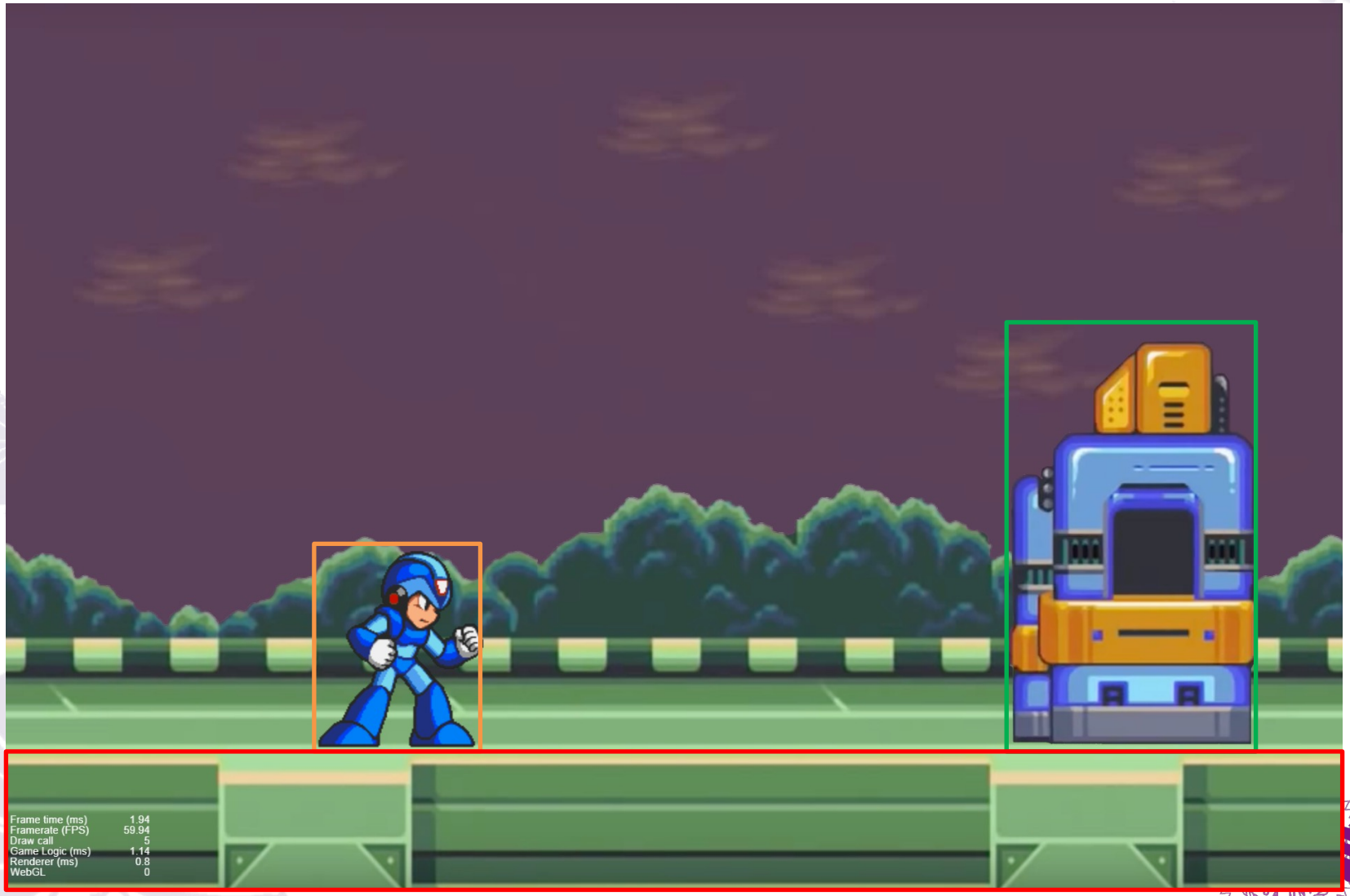
Player.ts

Let’s Play!


Frame time (ms) 0.69
Framerate (FPS) 59.99
Draw call 4
Game Logic (ms) 0.47
Renderer (ms) 0.22
WebGL 0

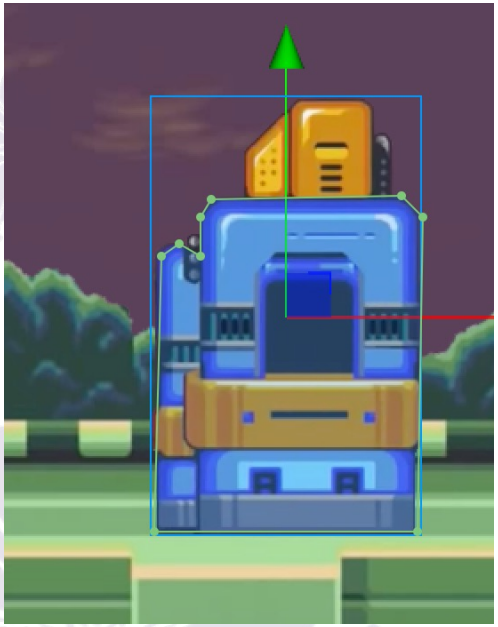

Take a
Break!
C++
PASCAL

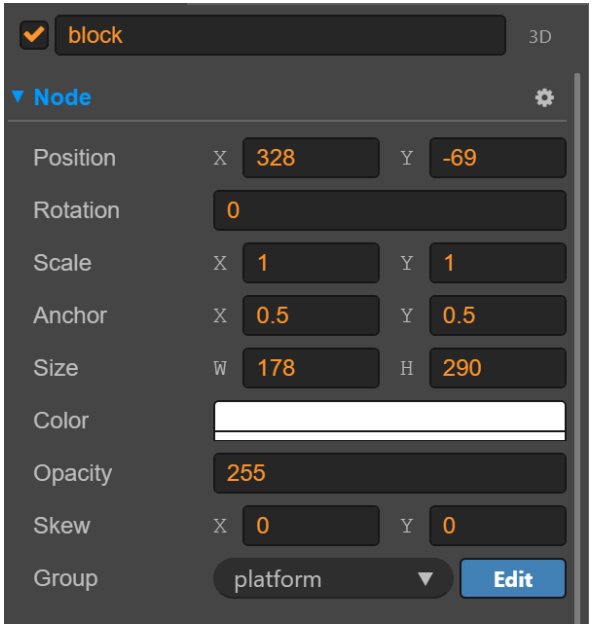
Add an Obstacle Node

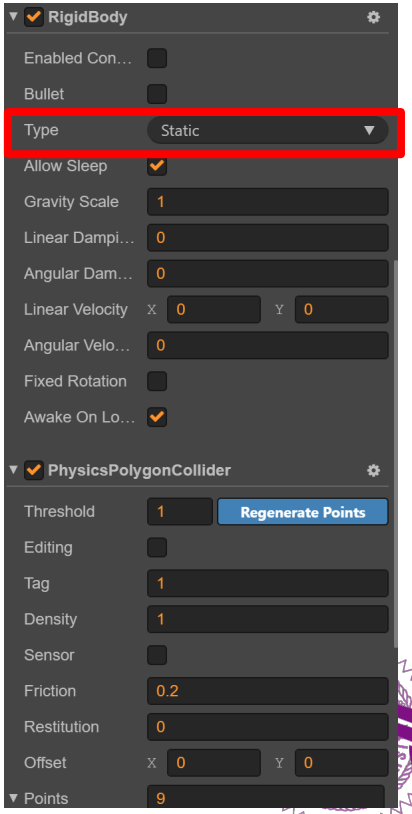


Basic Settings

- Add a new node named **block** with sprite
- Add a **Static RigidBody** & **Polygon Collider**

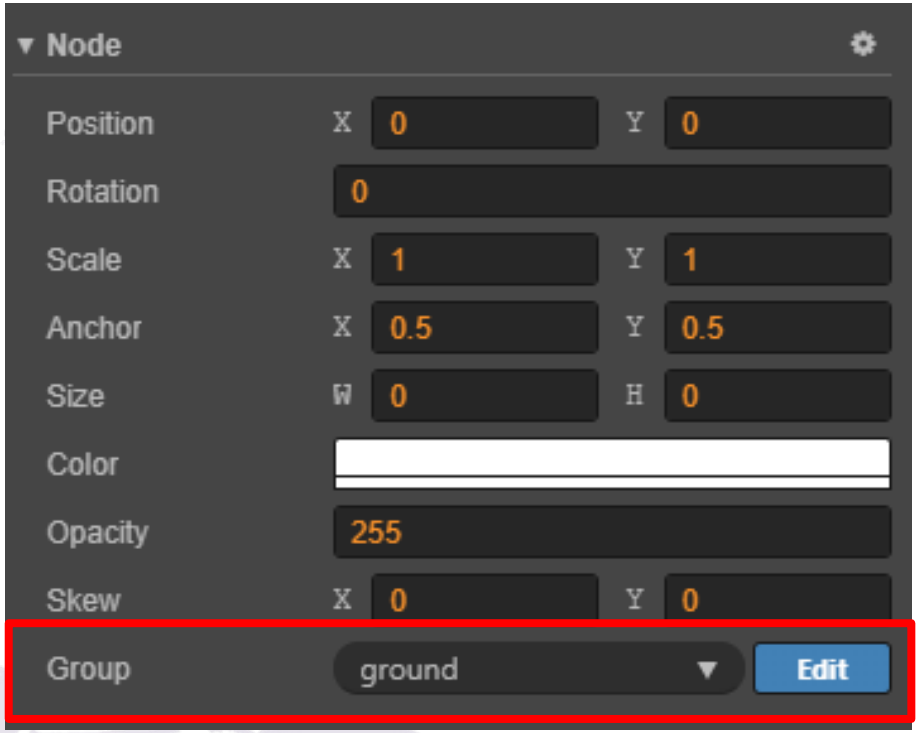


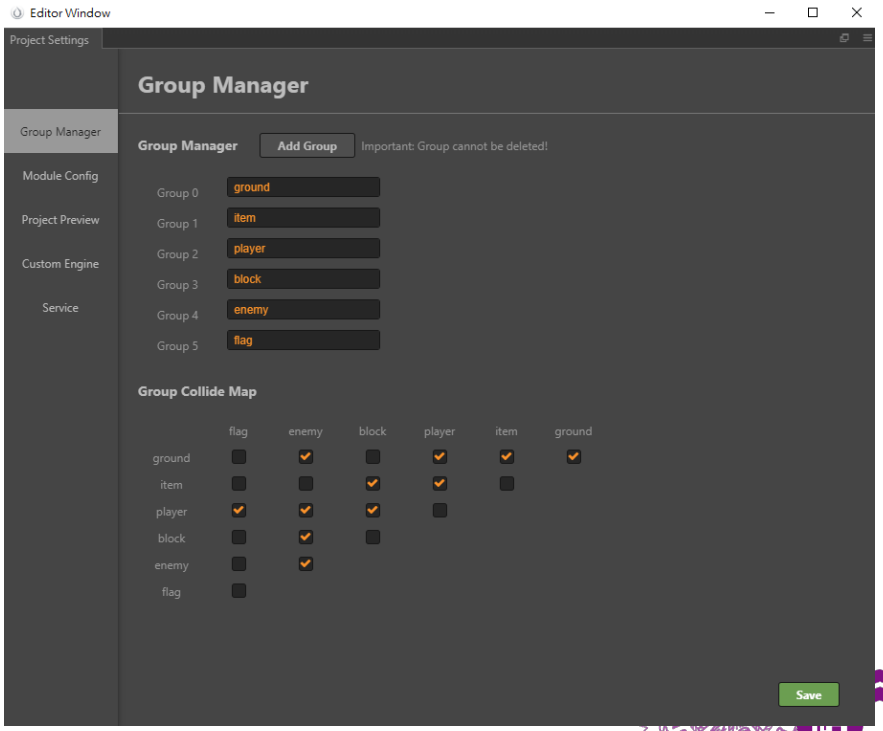




Group Manager

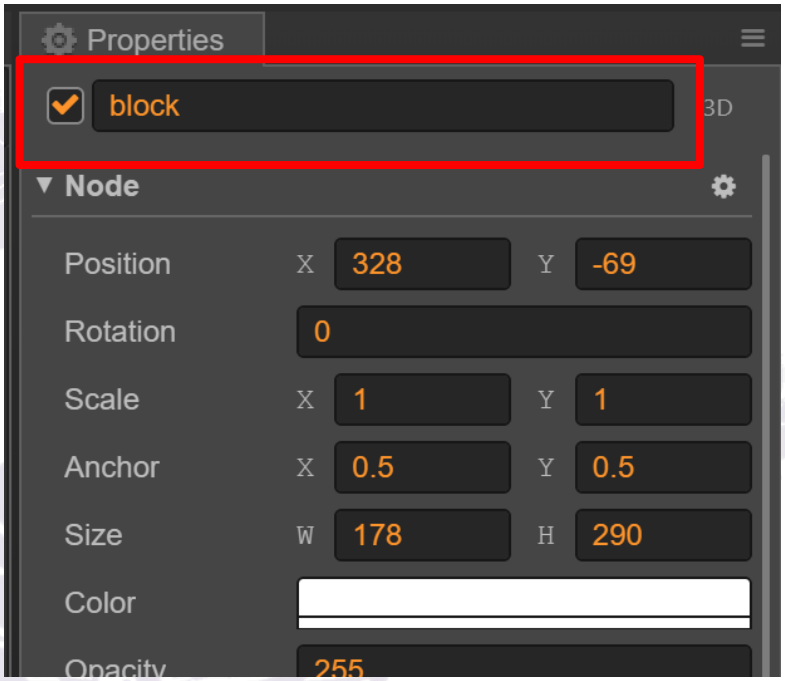
- We can use **Group Manager panel** to control which group can collide with which.

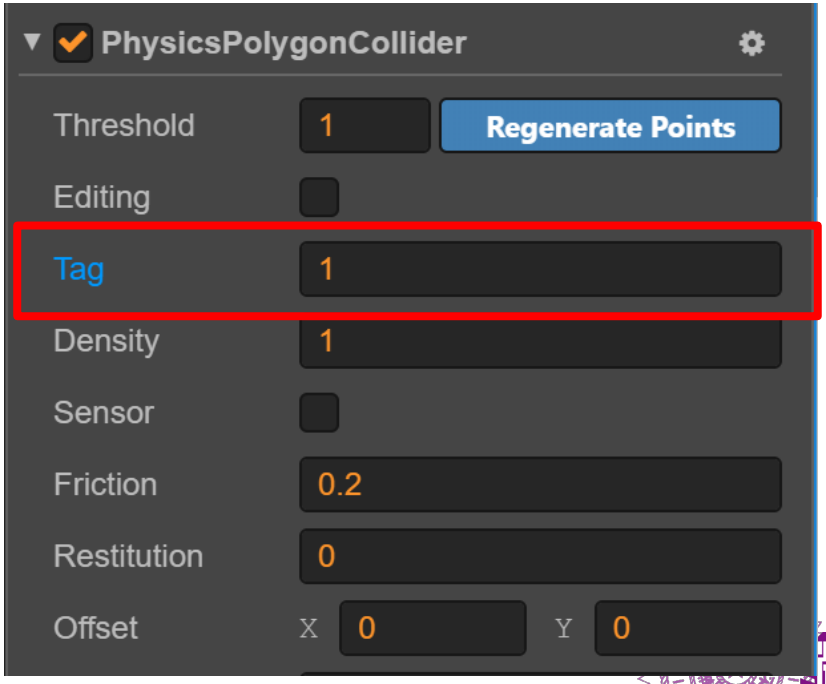




Identify the Colliding Node

- After editing collision group, We can use “**Name**” or “**Tag**” to identify which node the player collides with.

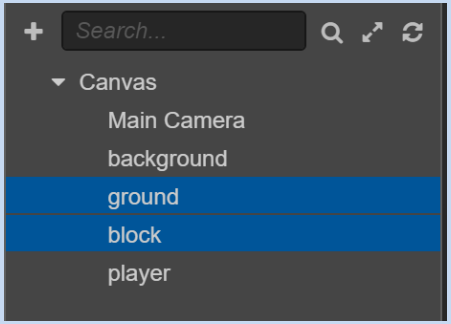




Identify the Colliding Node

- Identifying the colliding node via **Name.**

```
onLoad () {
        cc.director.getPhysicsManager().enabled = true;
}

onBeginContact(contact, self, other) {
        if(other.node.name == "ground") {
                cc.log("Rockman hits the ground");
        } else if(other.node.name == "block") {
                cc.log("Rockman hits the block");
        }

        this.onGround = true;
}
```

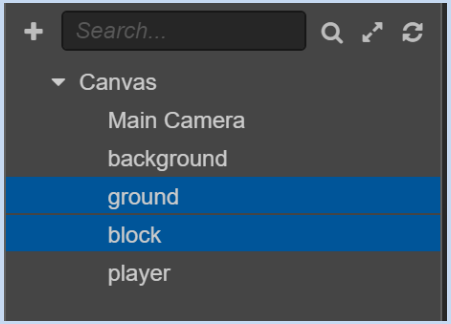


Player.ts

Identify the Colliding Node

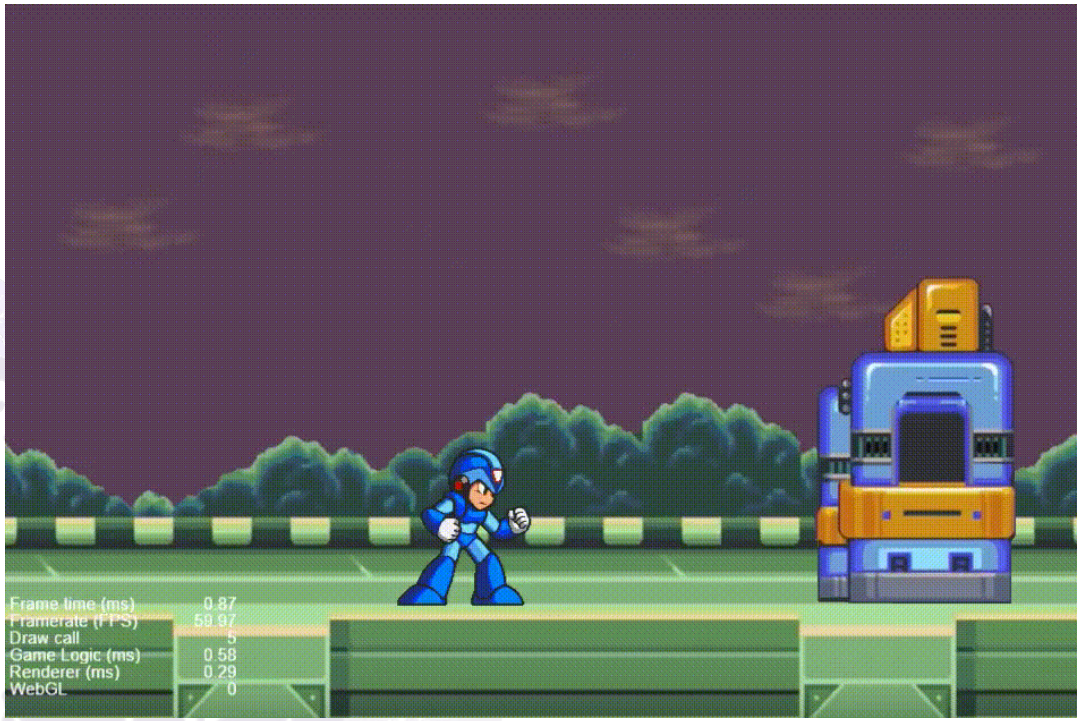
- Identifying the colliding node via **Tag.**

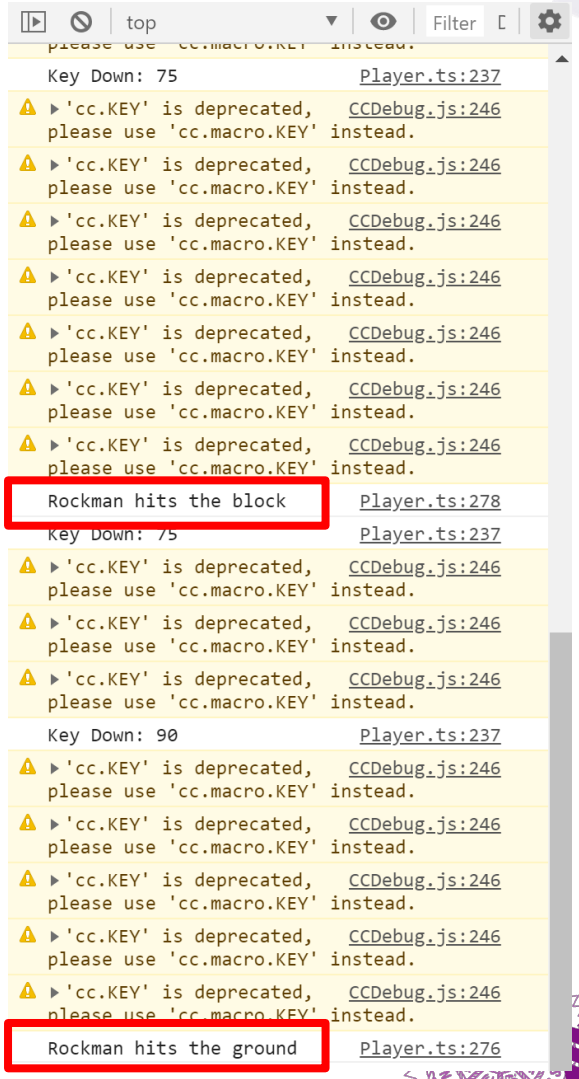
```
onLoad () {
        cc.director.getPhysicsManager().enabled = true;
}

onBeginContact(contact, self, other) {
        if(other.tag == 0) { // ground's tag
                cc.log("Rockman hits the ground");
        } else if(other.tag == 1) { // block's tag
                cc.log("Rockman hits the block");
        }

        this.onGround = true;
}
```



Player.ts

Identify the Colliding Node

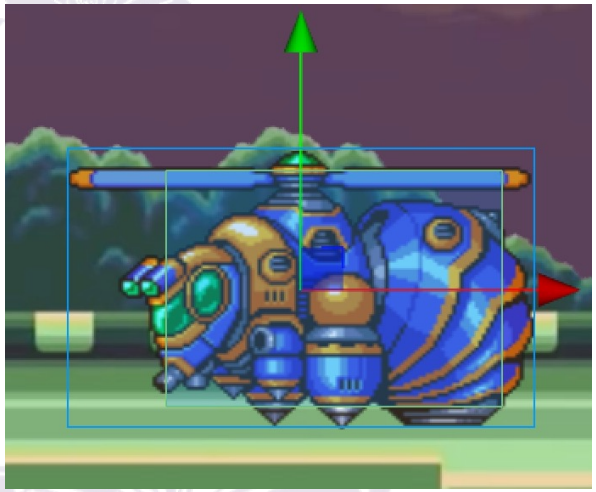


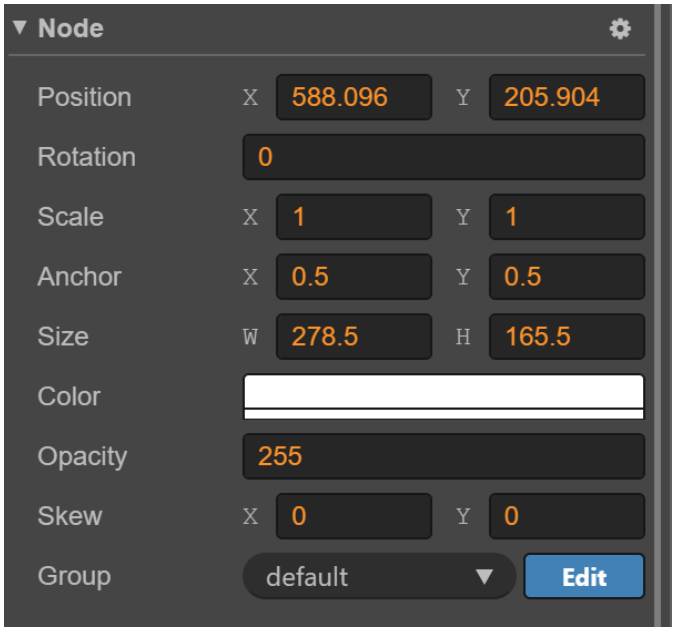


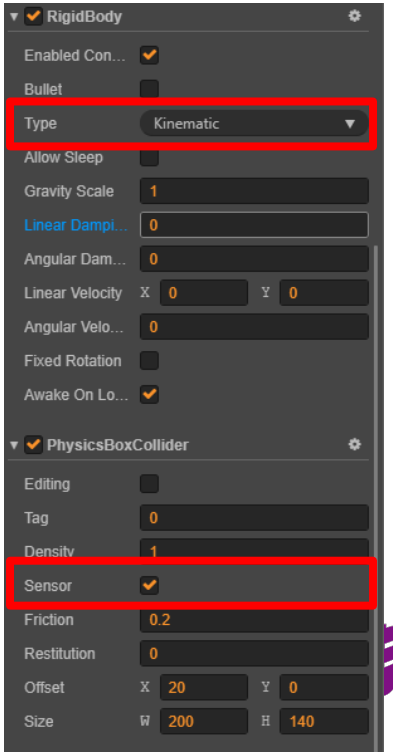
Add an Enemy Node

Basic Settings

- Add a new Node named **enemy** with sprite
- Add a **RigidBody** with **Kinematic** type
- Add a **Box Collider** with **Sensor** checked







Collider as a Trigger

- Detect which Node collide with Rockman

```
onBeginContact(contact, self, other) {
        if(other.node.name == "ground") {
                cc.log("Rockman hits the ground");
                this.onGround = true;
        }else if(other.node.name == "block") {
                cc.log("Rockman hits the block");
                this.onGround = true;
        }else if(other.node.name == "enemy") {
                cc.log("Rockman hits the enemy");
        }
}
```

Player.ts

Collide with Enemy

top | Filter

Key Down: 88 Player.ts:237

'cc.KEY' is deprecated, please use 'cc.macro.KEY' instead. CCDebug.js:246

'cc.KEY' is deprecated, please use 'cc.macro.KEY' instead. CCDebug.js:246

'cc.KEY' is deprecated, please use 'cc.macro.KEY' instead. CCDebug.js:246

Rockman hits the enemy Player.ts:284

Key Down: 88 Player.ts:237

'cc.KEY' is deprecated, please use 'cc.macro.KEY' instead. CCDebug.js:246

'cc.KEY' is deprecated, please use 'cc.macro.KEY' instead. CCDebug.js:246

'cc.KEY' is deprecated, please use 'cc.macro.KEY' instead. CCDebug.js:246

Key Down: 88 Player.ts:237

'cc.KEY' is deprecated, please use 'cc.macro.KEY' instead. CCDebug.js:246

'cc.KEY' is deprecated, please use 'cc.macro.KEY' instead. CCDebug.js:246

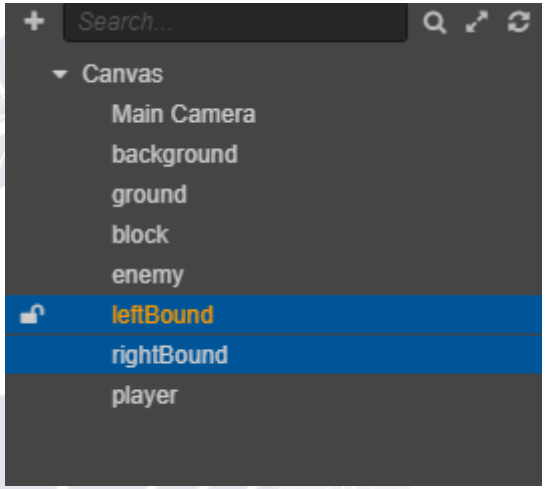
'cc.KEY' is deprecated, please use 'cc.macro.KEY' instead. CCDebug.js:246

Key Down: 88 Player.ts:237

Make a Bouncing Enemy

Add Left/Right Boundary

- To make enemy move back and forth in specified area, we add two nodes named **leftBound** and **rightBound**



Moving the Enemy

- Add new script named **Enemy** to enemy
- Make enemy change moving direction when it touches the left/right boundary

```
start() {
   this.node.getComponent(cc.RigidBody).linearVelocity = cc.v2(-100, 0);
}

onBeginContact(contact, self, other) {
   if(other.node.name == "leftBound") {
            // modify node's X scale value to change facing direction
            this.node.scaleX = -1;
            // modify rigidbody's velocity to make the node move
            this.node.getComponent(cc.RigidBody).linearVelocity = cc.v2(100, 0);
    } else if(other.node.name == "rightBound") {
            this.node.scaleX = 1;
            this.node.getComponent(cc.RigidBody).linearVelocity = cc.v2(-100, 0);
   }
}
```

Enemy.ts

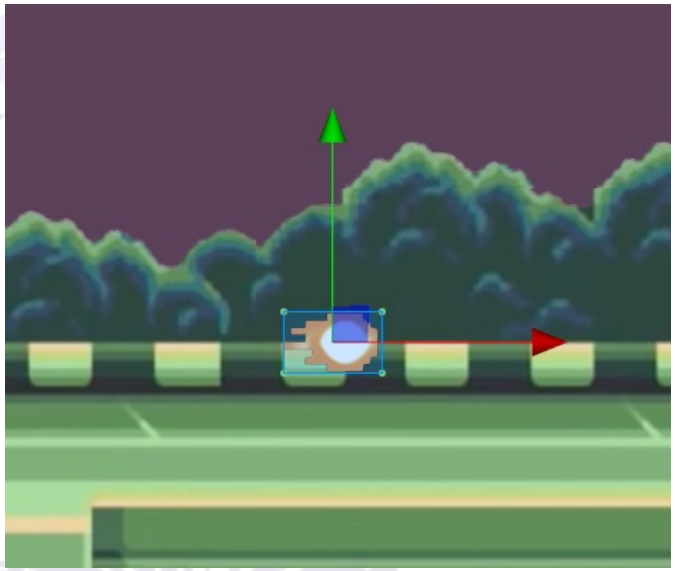
Let’s Play!

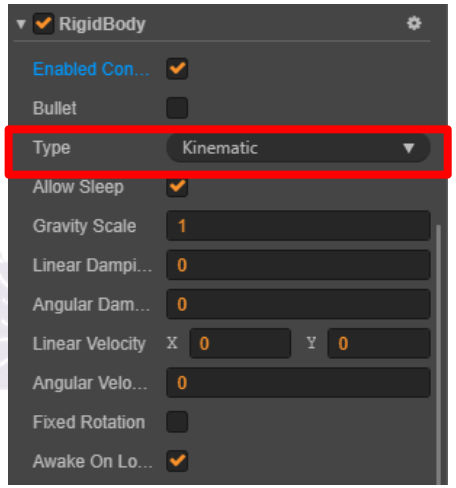

Frame time (ms) 1.23
Framerate (FPS) 60.05
Draw call 4
Game Logic (ms) 0.76
Renderer (ms) 0.46
WebGL 0

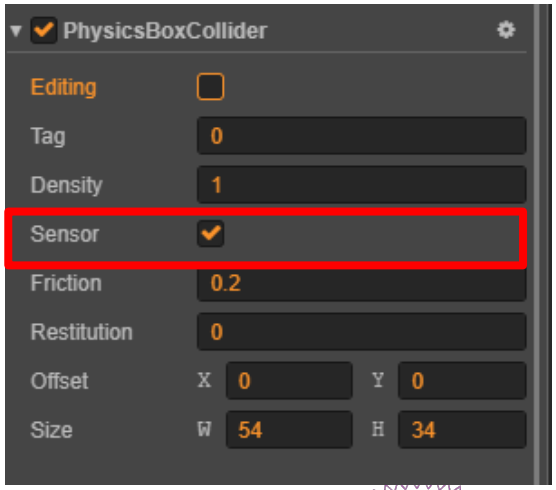
Add Bullets

Like Enemy Node

- Add a new Node named **bullet** with sprite
- Add a **RigidBody** with **Kinematic** type
- Add a **Box Collider** with **Sensor** checked







Bullet: Basic Settings

- Add a new script named **Bullet** to bullet
- Edit the script as follows:
  - Set initial position
  - Set RigidBody velocity
  - Destroy itself when it touches the boundary
- Edit the Player script to create bullet
- The bullet is shoot from the player's position; we need the player's node as input to initialize the bullet's position.

Bullet: Initial Position

```
public init(node: cc.Node) {
    this.setInitPos(node);
}

private setInitPos(node: cc.Node) {

// don't mount under the player, otherwise it will change direction when player move
    this.node.parent = node.parent;
    if(node.scaleX > 0) { // if the player is facing right
            // set the offset of bullet relative to center of rockman
            this.node.position = cc.v2(62, 8);
            this.node.scaleX = 1;
    } else {
            this.node.position = cc.v2(-62, 8);
            this.node.scaleX = -1;
    }
   // set the bullet's absolute position
   this.node.position = this.node.position.addSelf(node.position);
}
```

Bullet.ts

Bullet: RigidBody Velocity

```
public init(node: cc.Node) {
    this.setInitPos(node);
    this.bulletMove();
}
// set the bullet's velocity according to its moving direction.
private bulletMove() {
    let speed = 0;
    if(this.node.scaleX > 0) {
            speed = 600; // the speed of bullet
    } else {
            speed = -600;
    }
    // modify rigidbody's velocity to move the bullet
    this.node.getComponent(cc.RigidBody).linearVelocity = cc.v2(speed, 0);
}
```

Bullet.ts

Bullet: Destroy

```
public init(node: cc.Node) {
    this.setInitPos(node);
    this.bulletMove();
}
// set the bullet's velocity according to its moving direction.
private bulletMove() {
    let speed = 0;
    if(this.node.scaleX > 0) {
            speed = 600; // the speed of bullet
    } else {
            speed = -600;
    }
    // modify rigidbody's velocity to move the bullet
    this.node.getComponent(cc.RigidBody).linearVelocity = cc.v2(speed, 0);
}
```

Bullet.ts

Bullet: Destroy Itself

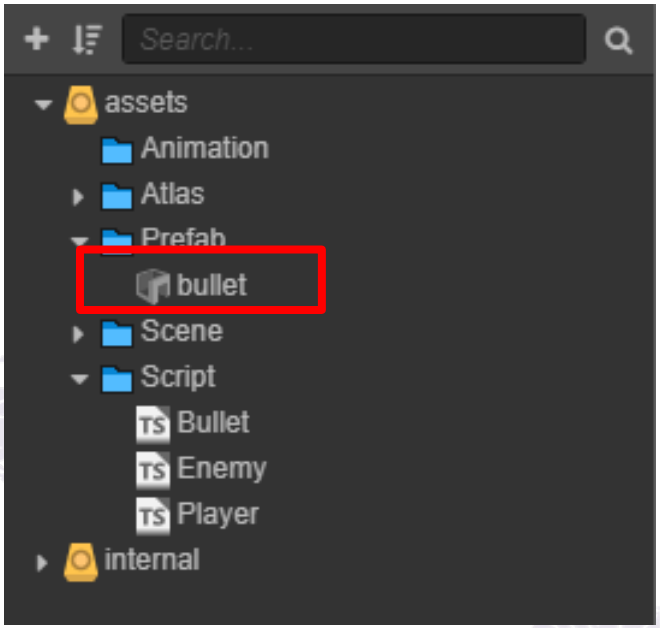
- Destroy the bullet when it is out of screen

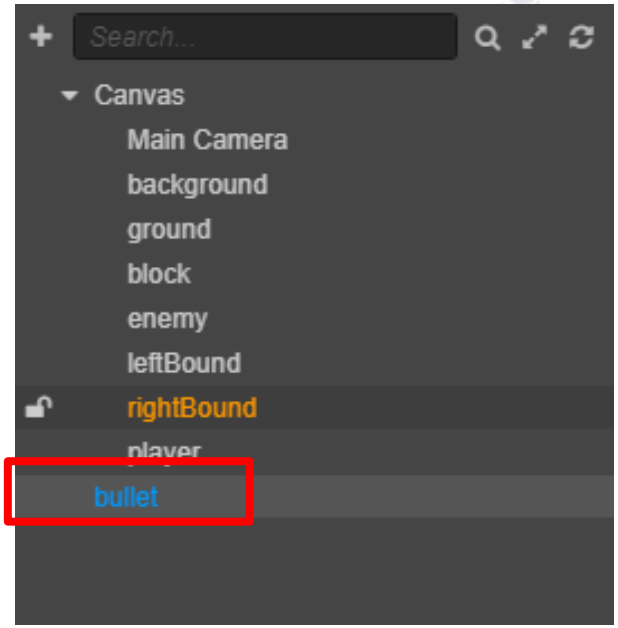
```
// destroy the bullet when it touches the boundary nodes
onBeginContact(contact, self, other) {
    if(other.node.name == "leftBound" || other.node.name == "rightBound") {
            self.node.destroy();
     }
}
```

Player: Fire a Bullet

- Make a bullet prefab
- Add bullet prefab as property
- Add keyboard event to fire a bullet
- New a bullet using **cc.instantiate(prefab)**

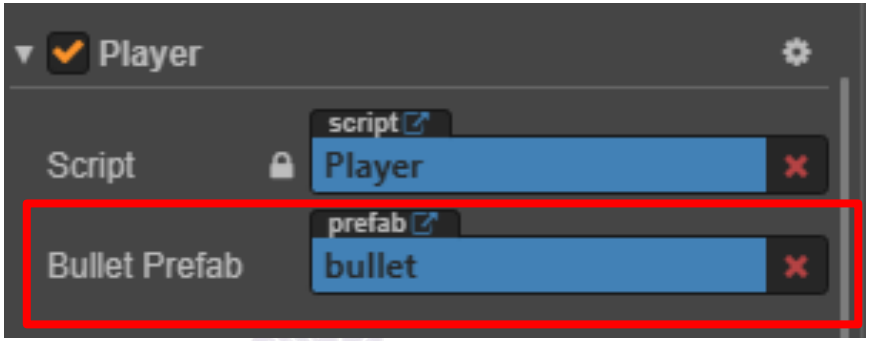
Make a Bullet Prefab





Add Bullet Prefab as Property

```
@property(cc.Prefab)
private bulletPrefab: cc.Prefab = null;
```



Keyboard Event

```
onKeyDown(event) {
    cc.log("Key Down: " + event.keyCode);
    if(event.keyCode == cc.macro.KEY.z) { // press key z to turn left
            this.zDown = true;
            this.xDown = false;
    } else if(event.keyCode == cc.macro.KEY.x) { // press key x to turn left
            this.xDown = true;
            this.zDown = false;
    } else if(event.keyCode == cc.macro.KEY.k) { // press key k to jump
            this.kDown = true;
    } else if(event.keyCode == cc.macro.KEY.j) { // press key j to fire
            this.jDown = true;
            this.createBullet();
    }
}
```

Player.ts

Create a Bullet Node using Prefab

```
private createBullet() {
        let bullet = cc.instantiate(this.bulletPrefab);
        bullet.getComponent('Bullet').init(this.node);
}
```

Player.ts

Let’s Play!


Frame time (ms) 0.8
Framerate (FPS) 60.03
Draw call 7
Game Logic (ms) 0.56
Renderer (ms) 0.24
WebGL 0

Enemy: Attack and Reborn

Enemy: Attack and Reborn

- When a bullet hits an enemy, we reset the enemy to a reborn point.
  - Note that the enemy acts as a **trigger** now.
- We destroy the bullet at the same moment.
- When the player touches an enemy, we reset the player to a reborn point.

Enemy: Reborn

```
start() {
    // get initial position and set initial velocity
    this.rebornPos = this.node.position;
    this.node.getComponent(cc.RigidBody).linearVelocity = cc.v2(-100, 0);
    this.isDead = false;
}
onBeginContact(contact, self, other) {
    if(other.node.name == "leftBound") {
            this.node.scaleX = -1;
            this.node.getComponent(cc.RigidBody).linearVelocity = cc.v2(100, 0);
     } else if(other.node.name == "rightBound") {
            this.node.scaleX = 1;
            this.node.getComponent(cc.RigidBody).linearVelocity = cc.v2(-100, 0);
     } else if(other.node.name == "bullet") {
             // we don't physically remove the enemy but reset its position instead
            this.isDead = true;
            other.node.destroy();  // destroy the bullet physically
    }
}
```

Enemy.ts

Enemy: Reborn

```
public resetPos() {
    this.node.position = this.rebornPos;
    this.node.scaleX = 1;
    this.node.getComponent(cc.RigidBody).linearVelocity = cc.v2(-100, 0);
}

update(dt) {
    if(this.isDead) {
            this.resetPos(); // reset the enemy's position when it dies
            this.isDead = false;
    }
}
```

Enemy.ts

Player: Reborn

```
onBeginContact(contact, self, other) {
        if(other.node.name == "ground") {
                cc.log("Rockman hits the ground");
                this.onGround = true;
        }else if(other.node.name == "block") {
                cc.log("Rockman hits the block");
                this.onGround = true;
        }else if(other.node.name == "enemy") {
                cc.log("Rockman hits the enemy");
                this.isDead = true;
        }
}
```

Player.ts

Player: Reborn (Cont'd)

```
private playerMovement(dt) {
   this.playerSpeed = 0;
   if(this.isDead) {
            // reset the player's velocity to zero and set its position to a reborn position.
            this.node.getComponent(cc.RigidBody).linearVelocity = cc.v2(0, 0);
            this.node.position = cc.v2(-192, 255);
            this.isDead = false;
            return;
    }
    if(this.zDown) {
            this.playerSpeed = -300;
            this.node.scaleX = -1;
    } else if(this.xDown) {
            this.playerSpeed = 300;
            this.node.scaleX = 1;
    }
    this.node.x += this.playerSpeed * dt;

    if(this.kDown && this.onGround)  this.jump();
}
```

Player.ts

Let’s Play!


Frame time (ms) 0.68
Framerate (FPS) 59.86
Draw call 7
Game Logic (ms) 0.46
Renderer (ms) 0.22
WebGL 0


thank you!
Question?