Software Studio
軟體設計與實驗

Slime AI Practice

Hung-Kuo Chu
Department of Computer Science
National Tsing Hua University

CS2410

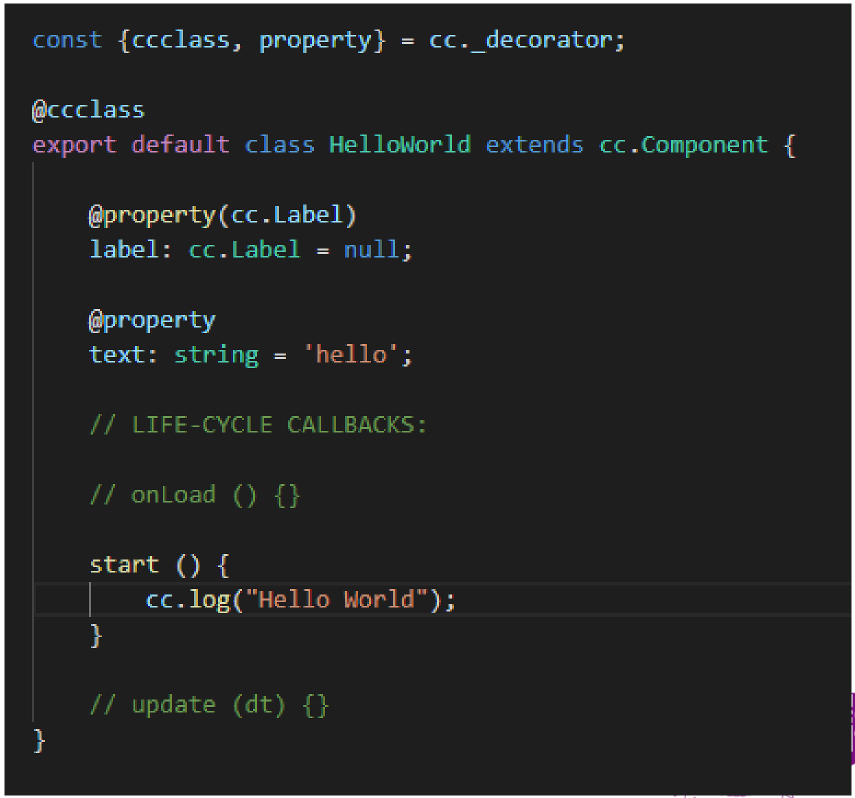
Slime AI

- We will create the slime AI in this project
- You will learn the concept of OOP and hierarchy, and some basic “**script**” technique in this practice.

Script: Basics

- Cocos Creator supports language:
  - **Typescript**, JavaScript, CoffeeScript

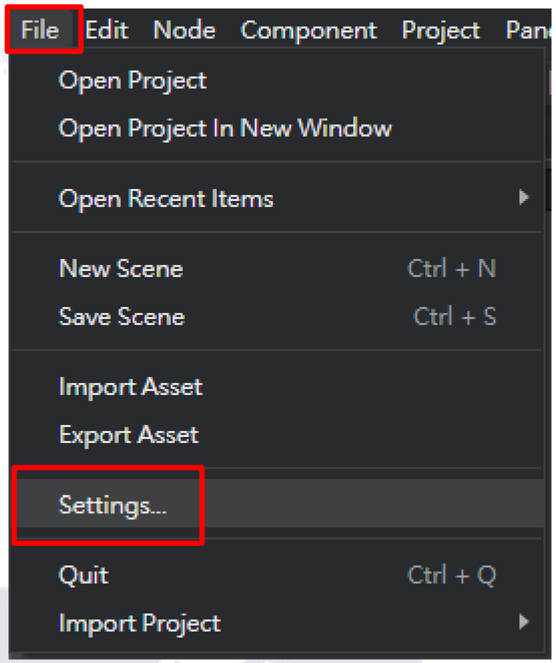
- Recommended IDE
  - Visual Studio Code

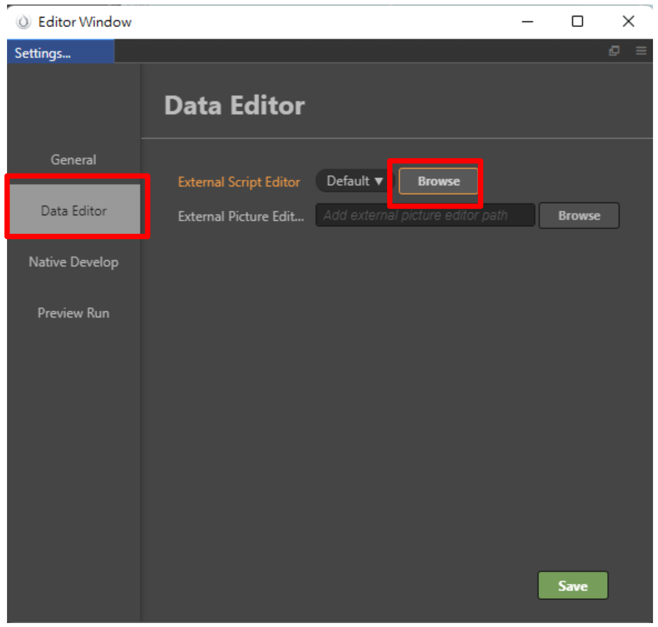


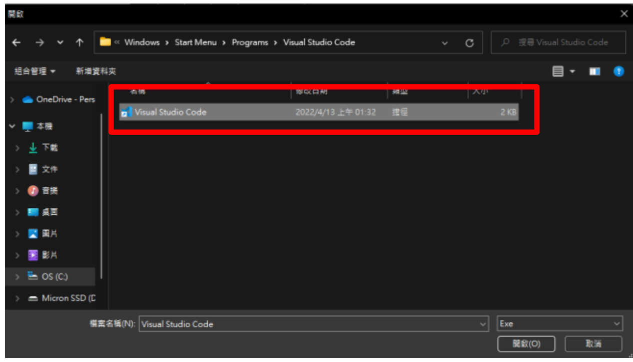
```
const {ccclass, property} = cc._decorator;

@ccclass
export default class HelloWorld extends cc.Component {

    @property(cc.Label)
    label: cc.Label = null;

    @property
    text: string = 'hello';

    // LIFE-CYCLE CALLBACKS:

    // onLoad () {}

    start () {
        cc.log("Hello World");
    }

    // update (dt) {}
}
```

Environment Setting (Windows)

- Choose default IDE editor
  - File → Settings → Data Editor →External Script Editor → Browse →Choose your IDE

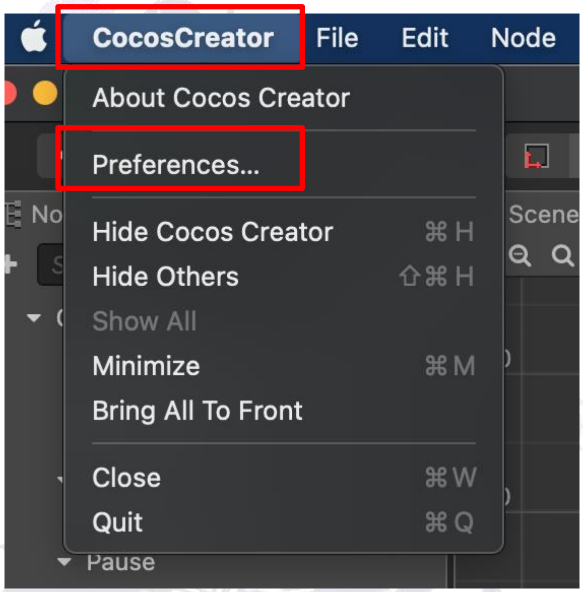


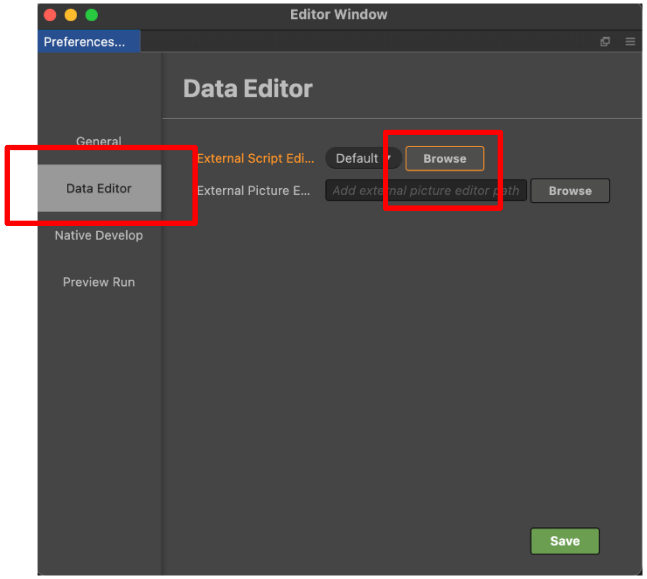


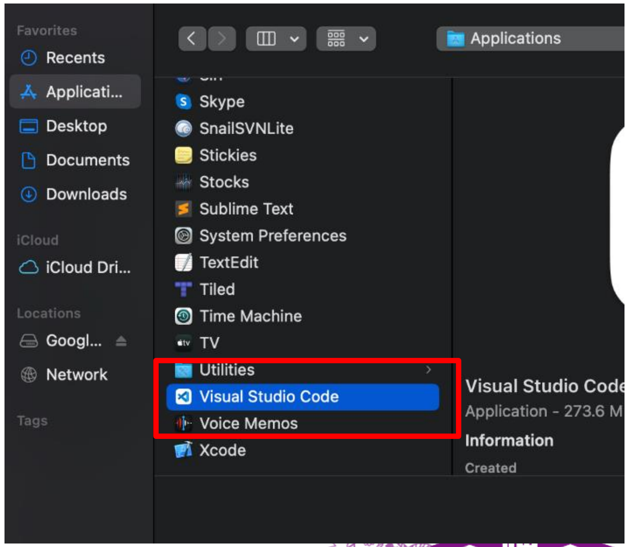


Environment Setting (MacOS)

- Choose default IDE editor
  - Cocoscreator → Preferences → Data Editor →External Script Editor → Browse → Choose your IDE

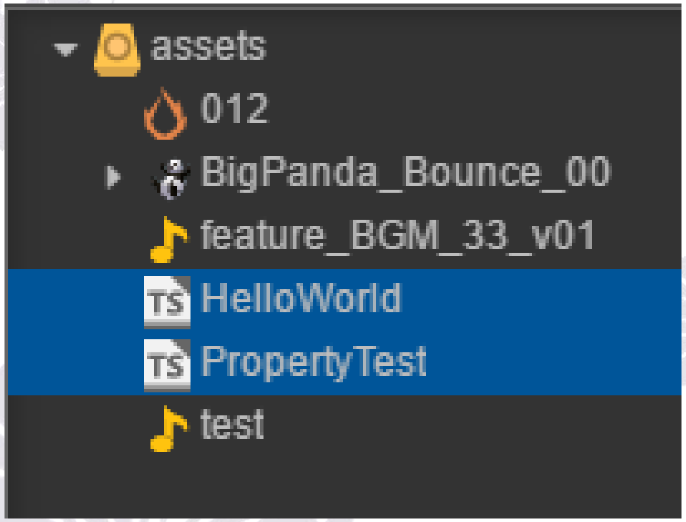


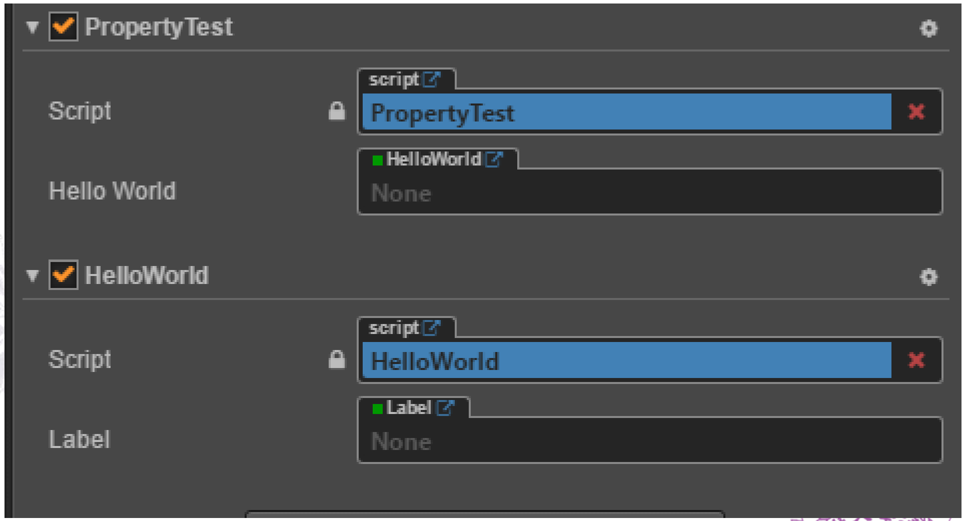




How does Script do?

- Control the behaviors of the Node
- Get information from the Node
- Run your own component





PropertyTest

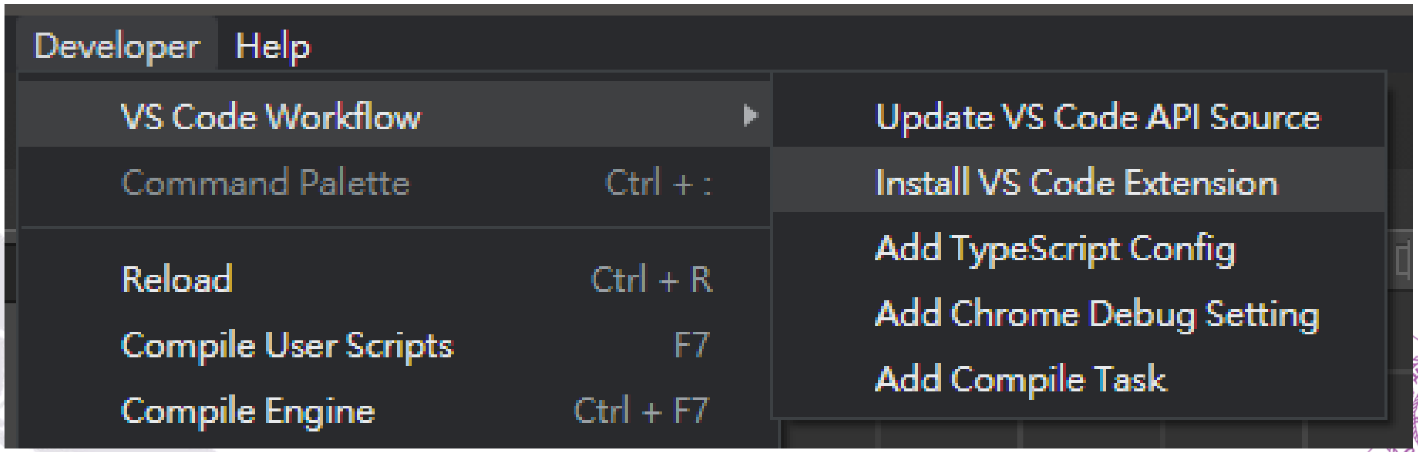
Script	PropertyTest
Hello World	None

HelloWorld

Script	HelloWorld
Label	None

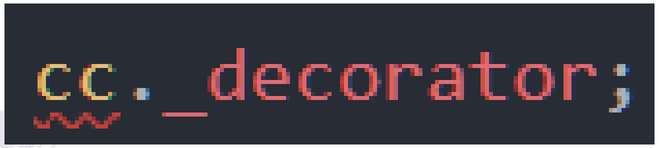
VS Code Extension

- You will only need to change the file in "assets/script" in the practice
- You can install VS code extension through Developer->VS code Workflow->Install VS Code Extension

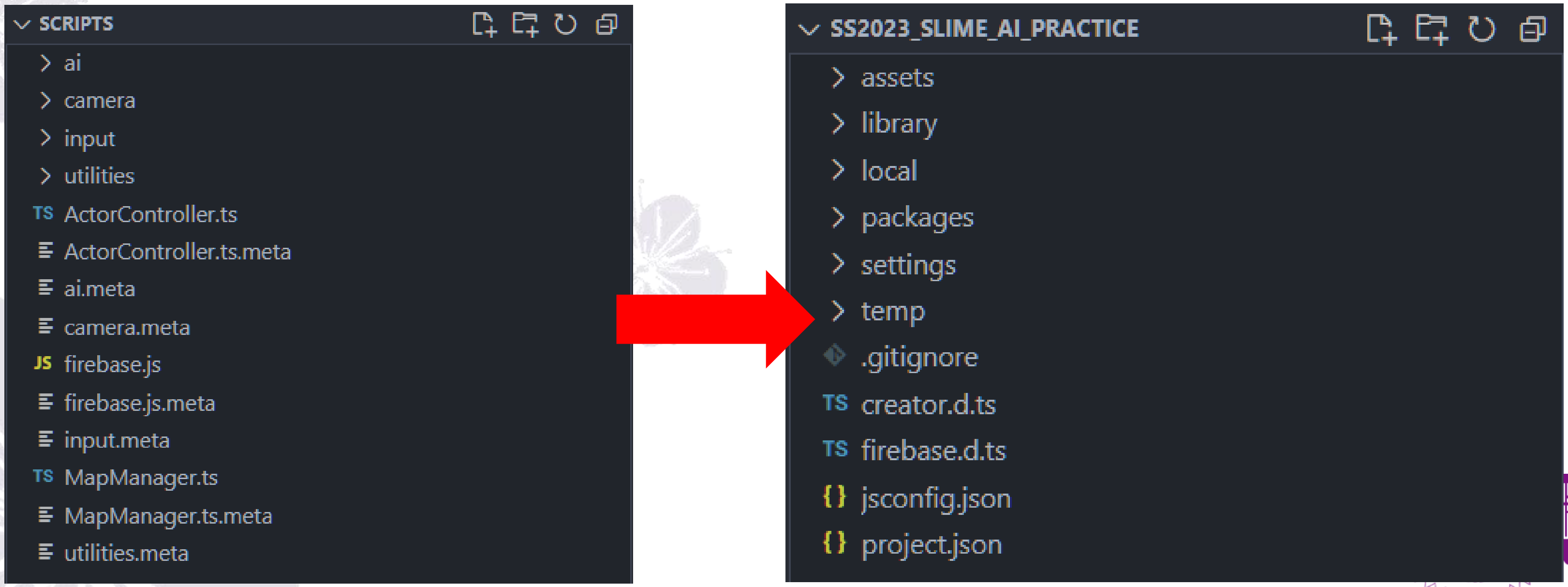


cc. error

- If you have this error, you need to open the whole project through VS code, not only the "script" directory.



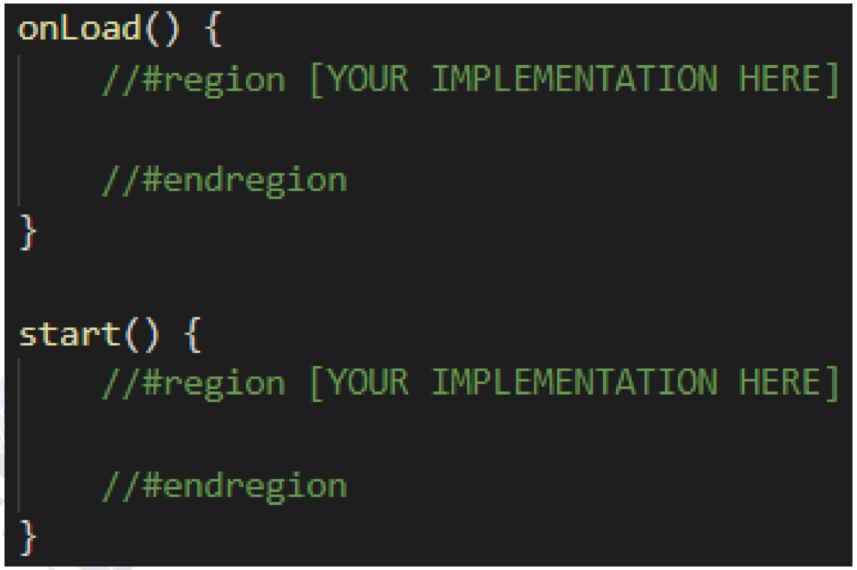
```
cc._decorator;
```



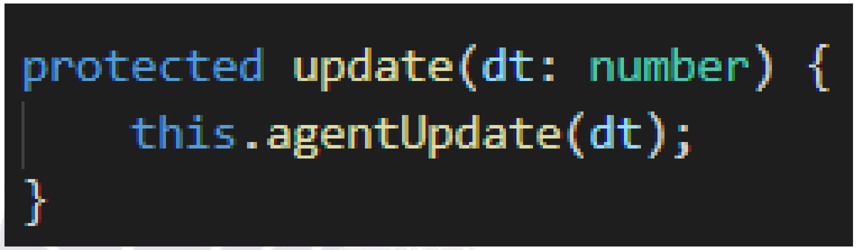
```
SCRIPTS
  > ai
  > camera
  > input
  > utilities
  ActorController.ts
  ActorController.ts.meta
  ai.meta
  camera.meta
  firebase.js
  firebase.js.meta
  input.meta
  MapManager.ts
  MapManager.ts.meta
  utilities.meta
```

```
SS2023_SLIME_AI_PRACTICE
  > assets
  > library
  > local
  > packages
  > settings
  > temp
  .gitignore
  creator.d.ts
  firebase.d.ts
  jsconfig.json
  project.json
```

Script: Life-Cycle Callbacks



```
onLoad() {
    //#region [YOUR IMPLEMENTATION HERE]

    //#endregion
}

start() {
    //#region [YOUR IMPLEMENTATION HERE]

    //#endregion
}
```

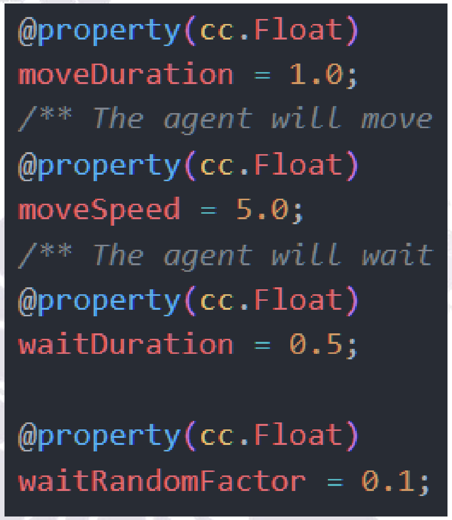


```
protected update(dt: number) {
    this.agentUpdate(dt);
}
```

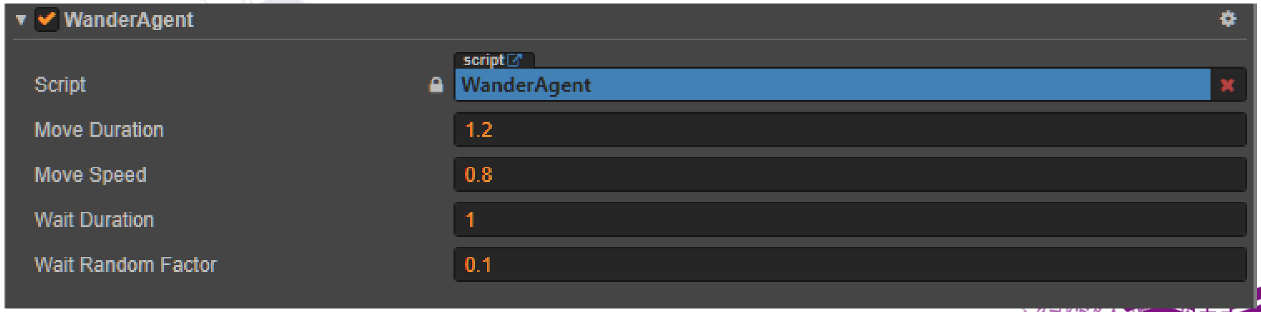
- **onLoad():** Run the code when the game start.
- **start():** Run the code after **all component** finish onLoad().
- **update(dt: number):** Called **every frame** in the game. **dt** is time passed since last frame.

Script: @property

- You can use **@property** to decorate variable as **property** in your script, then you can set the value of property in cocos IDE.
- For example, in **WanderAgent.ts**, we define some properties through **@property**, then you can directly set these properties in cocos IDE.



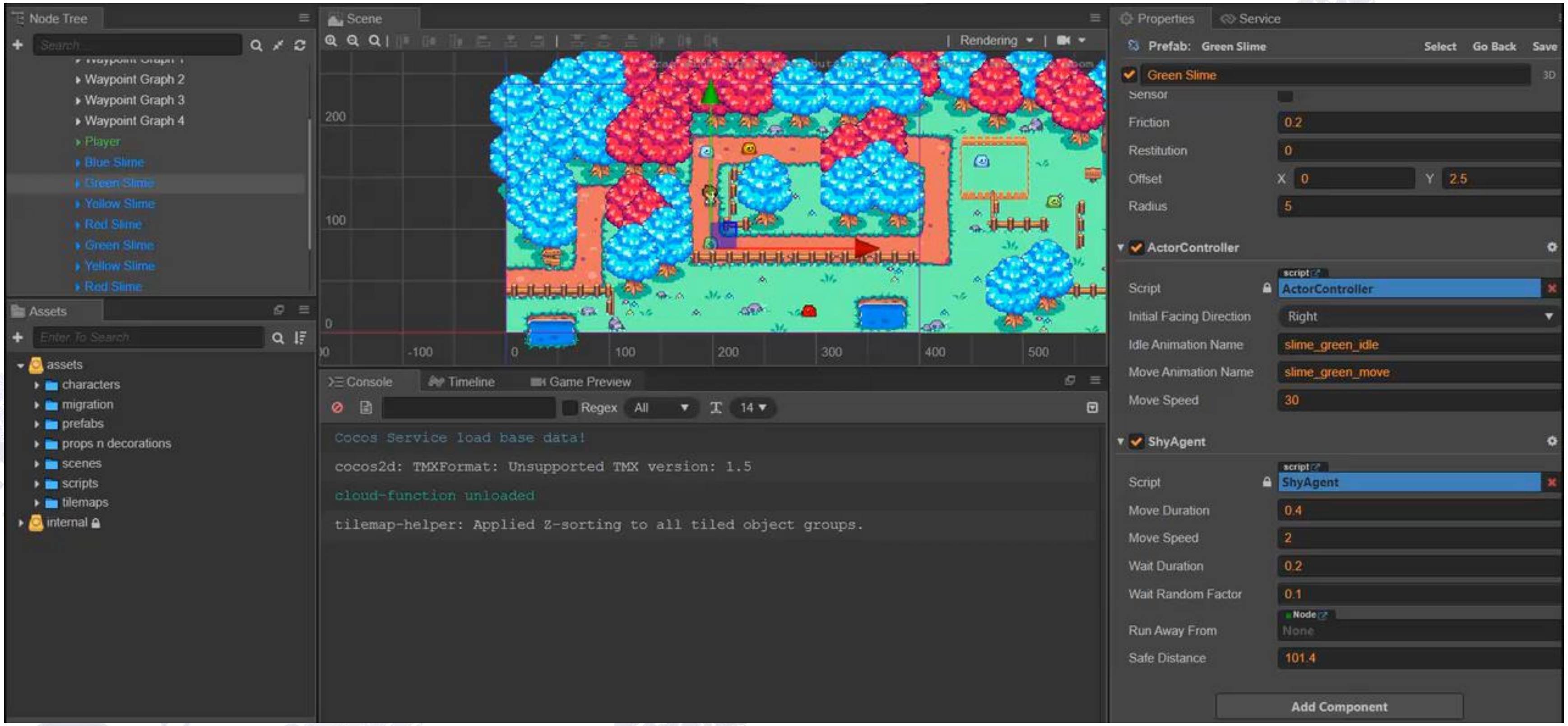
```
@property(cc.Float)
moveDuration = 1.0;
/** The agent will move
@property(cc.Float)
moveSpeed = 5.0;
/** The agent will wait
@property(cc.Float)
waitDuration = 0.5;

@property(cc.Float)
waitRandomFactor = 0.1;
```



WanderAgent

Script	WanderAgent
Move Duration	1.2
Move Speed	0.8
Wait Duration	1
Wait Random Factor	0.1

Script: Node Assignment



```
Cocos Service load base data!
cocos2d: TMXFormat: Unsupported TMX version: 1.5
cloud-function unloaded
tilemap-helper: Applied Z-sorting to all tiled object groups.
```

Prefab: Green Slime

Green Slime

Sensor	
Friction	0.2
Restitution	0
Offset	X 0 Y 2.5
Radius	5

ActorController

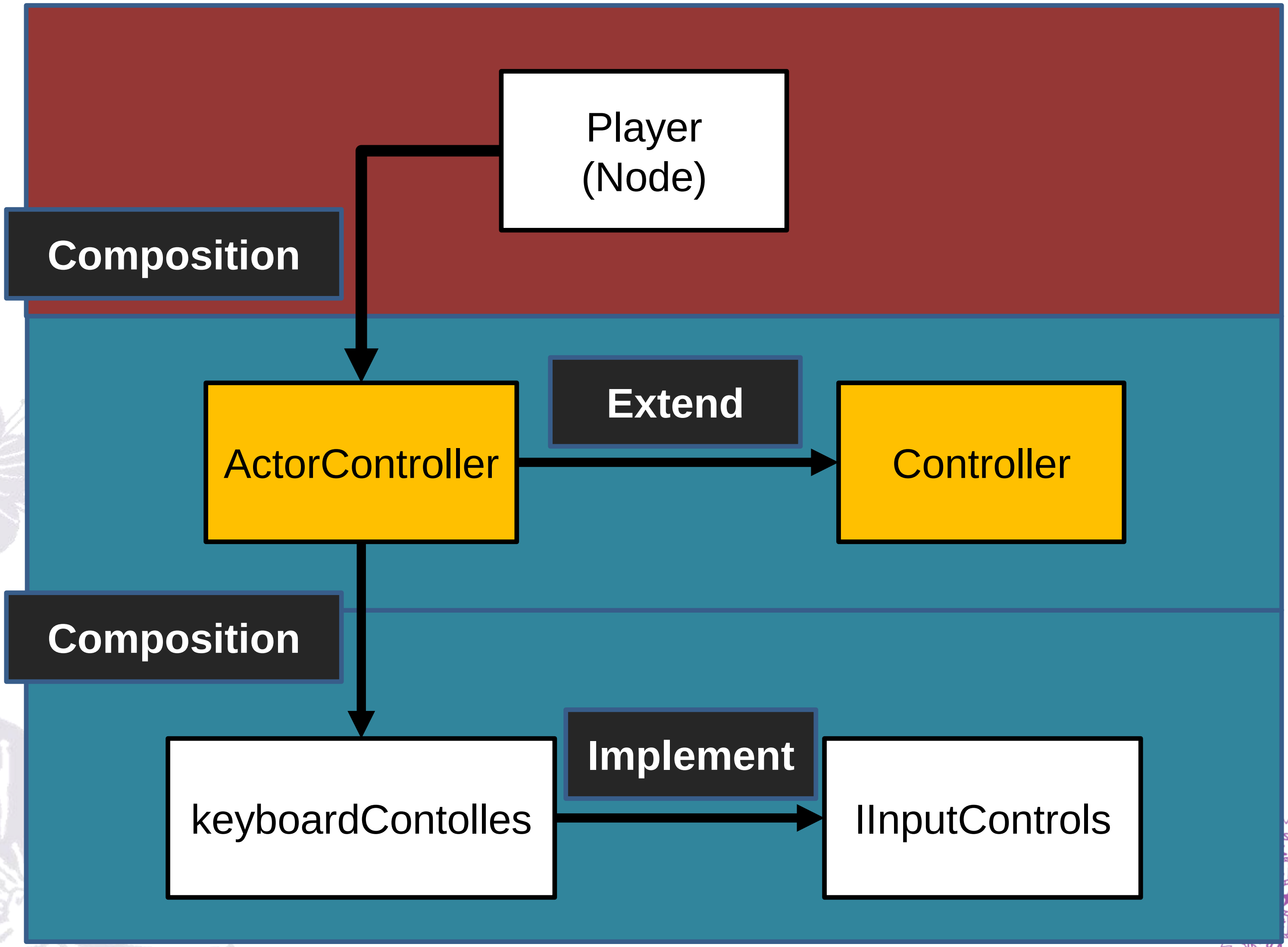
Script	ActorController
Initial Facing Direction	Right
Idle Animation Name	slime_green_idle
Move Animation Name	slime_green_move
Move Speed	30

ShyAgent

Script	ShyAgent
Move Duration	0.4
Move Speed	2
Wait Duration	0.2
Wait Random Factor	0.1
Run Away From	None
Safe Distance	101.4

You can drag a node that has a certain component into the inspector to assign a **reference** to that component.

Player: Hierarchy

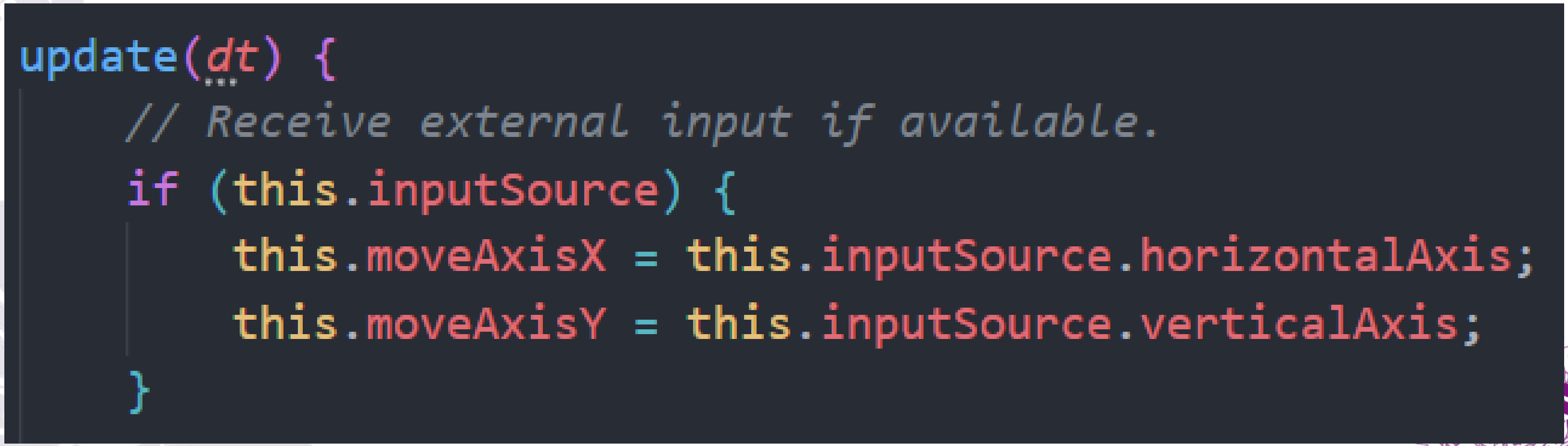


TODO 1

- Files: **keyboardControl.ts**(1.1, 1.2)
- In this TODO, we need to add controller to character, so the character can move up, down, left, right.
- Goal: **Use WASD keys to move character.**

ActorController.ts

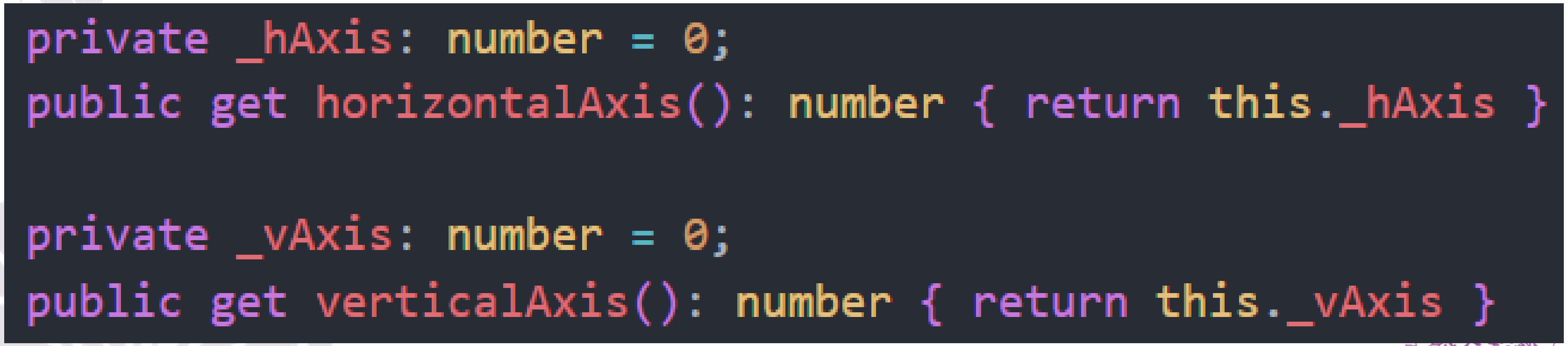
- In this script, we use **horizontalAxis** and **verticalAxis** methods to get object’s movement in X (**moveAxisX**) and Y (**moveAxisY**) directions in the update().
- Find the module that implements these two methods! -> **keyboardController.ts!**



```
update(dt) {
    // Receive external input if available.
    if (this.inputSource) {
        this.moveAxisX = this.inputSource.horizontalAxis;
        this.moveAxisY = this.inputSource.verticalAxis;
    }
```

TODO 1.1

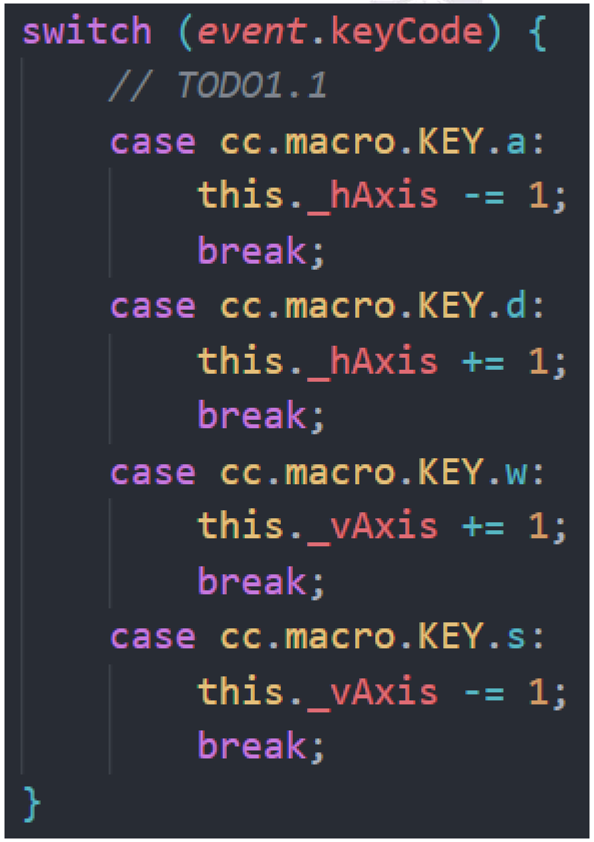
- In **keyboardController.ts**, we implement two interface (IInputControls) methods, **horizontalAxis()** and **verticalAxis(),** by returning two variables, **_hAxis** and **_vAxis**, respectively.



```
private _hAxis: number = 0;
public get horizontalAxis(): number { return this._hAxis }

private _vAxis: number = 0;
public get verticalAxis(): number { return this._vAxis }
```

TODO 1.1

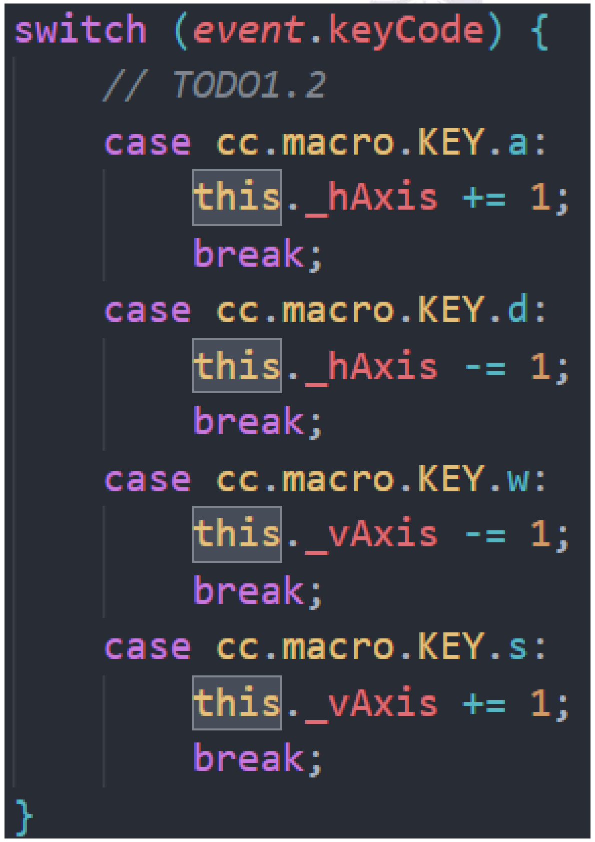
- In the **onKeyDown** method, use keycodes to move character left (a), right (d), up (w) and down (s).



```
switch (event.keyCode) {
    // TODO1.1
    case cc.macro.KEY.a:
        this._hAxis -= 1;
        break;
    case cc.macro.KEY.d:
        this._hAxis += 1;
        break;
    case cc.macro.KEY.w:
        this._vAxis += 1;
        break;
    case cc.macro.KEY.s:
        this._vAxis -= 1;
        break;
}
```

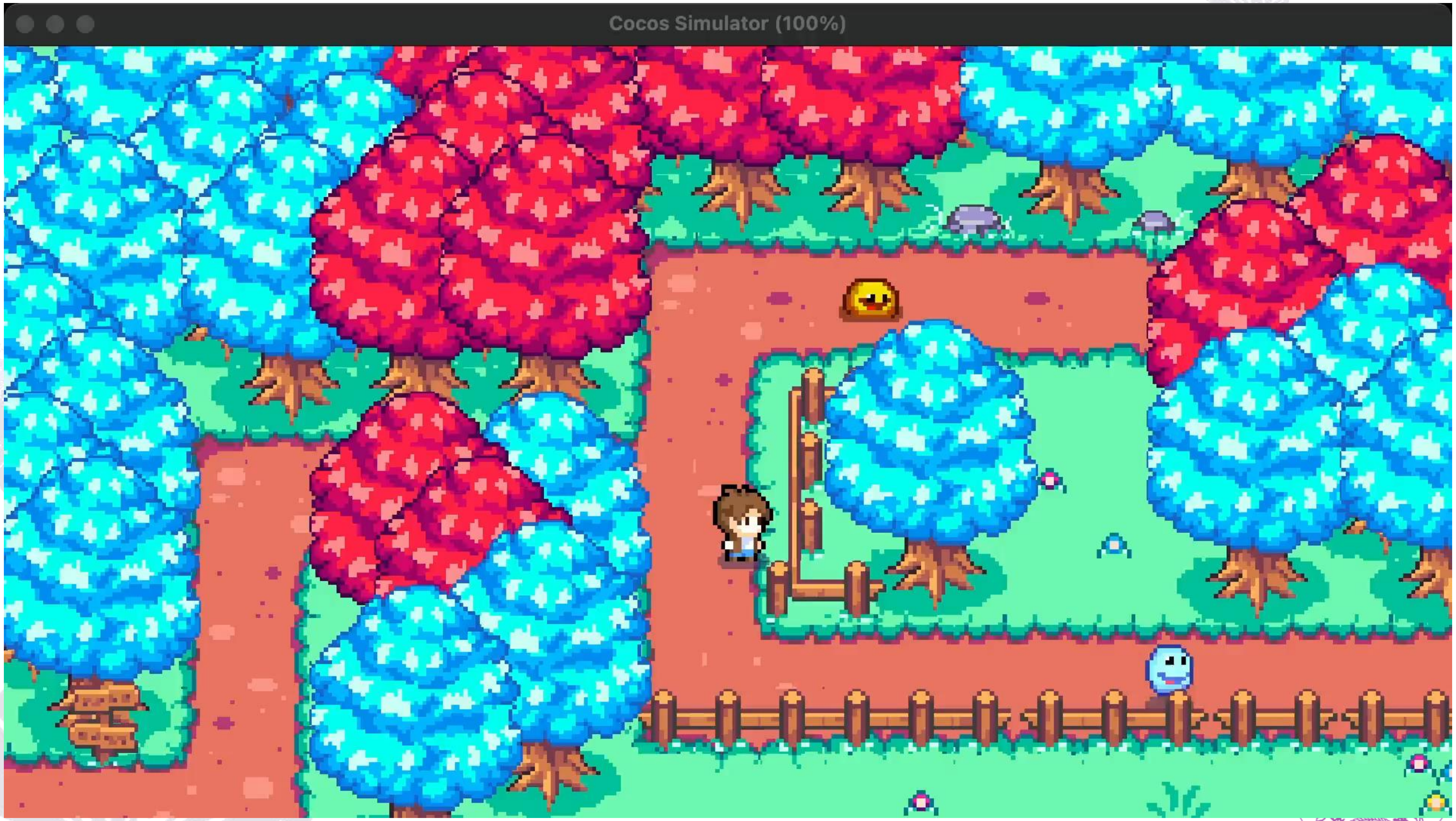
TODO 1.1

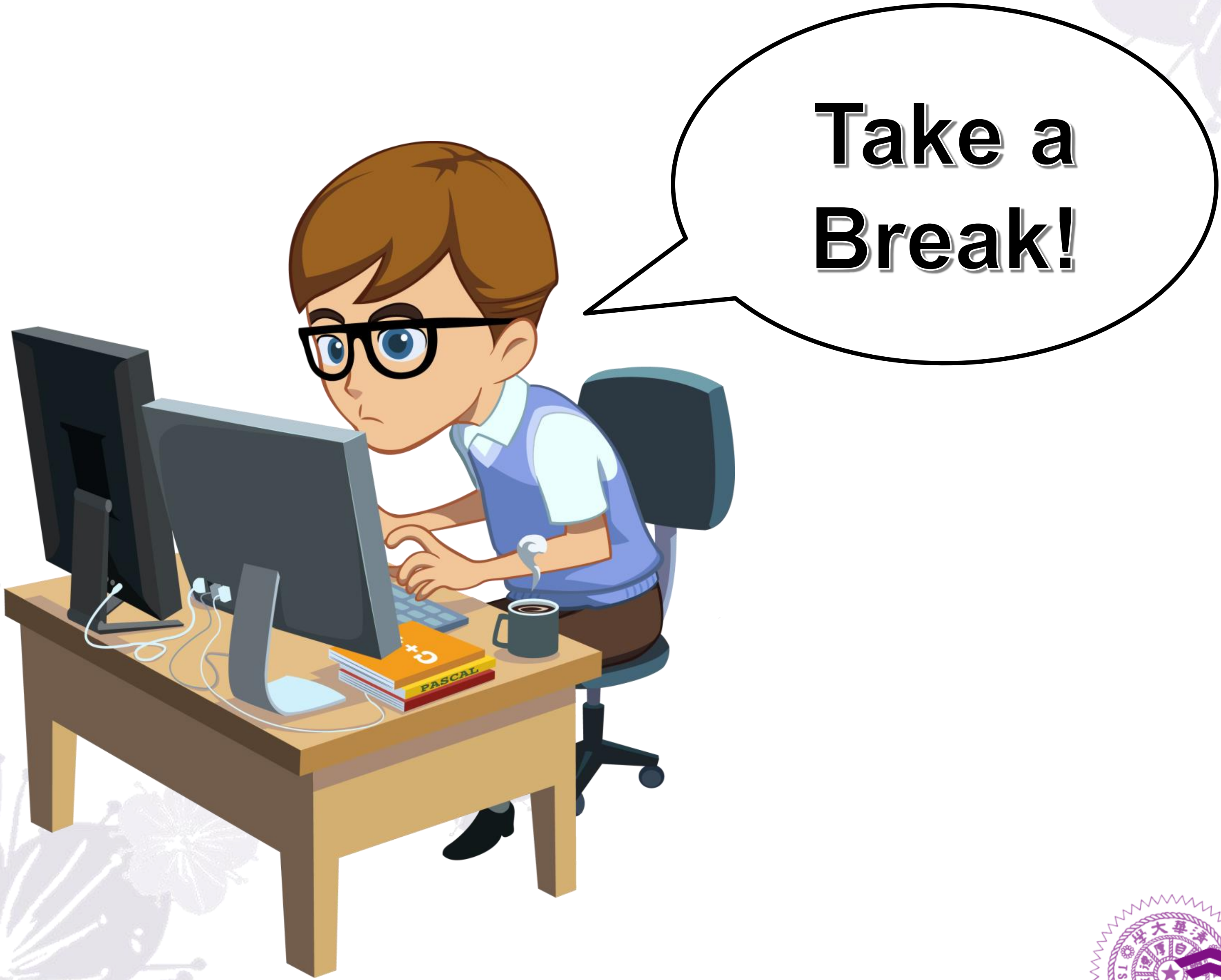
- In the **onKeyUp** method, we do the opposite way as we did in onKeyDown method.
- If D key is released, _hAxis should decrease by 1, so the character can stop moving right by 1 unit in each frame.



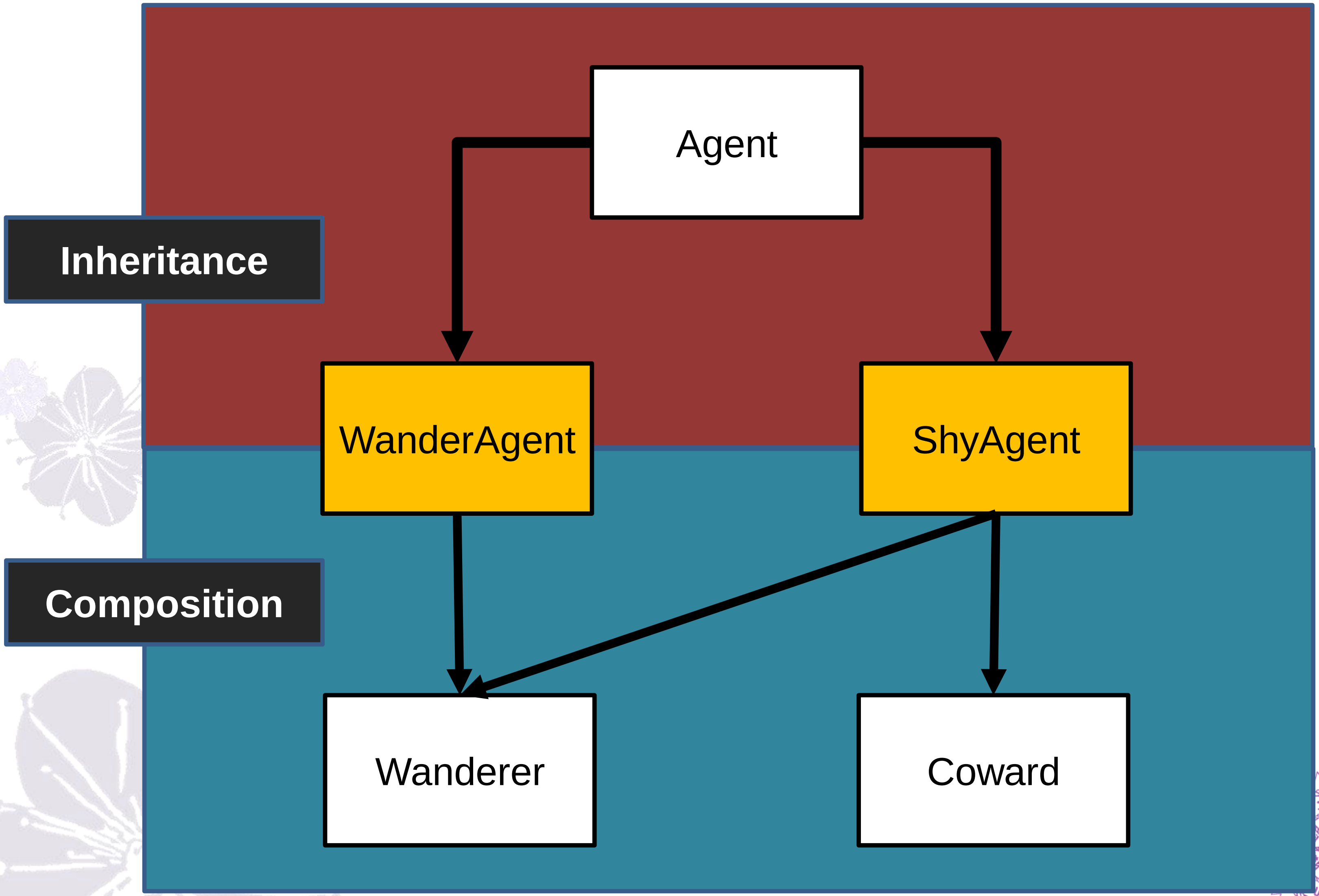
```
switch (event.keyCode) {
    // TODO1.2
    case cc.macro.KEY.a:
        this._hAxis += 1;
        break;
    case cc.macro.KEY.d:
        this._hAxis -= 1;
        break;
    case cc.macro.KEY.w:
        this._vAxis -= 1;
        break;
    case cc.macro.KEY.s:
        this._vAxis += 1;
        break;
}
```

Result




Take a Break!
C++
PASCAL

Agent: Hierarchy

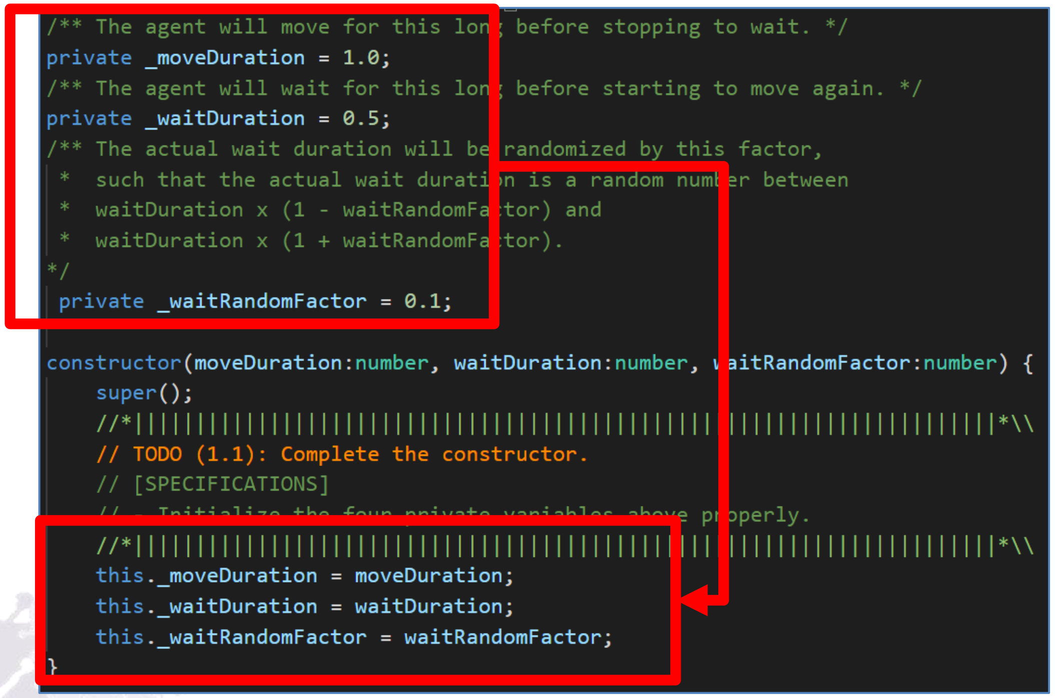


TODO 2

- Files: **Wanderer.ts (2.1, 2.2, 2.3), WanderAgent.ts (2.4)**
- Follow the hints in these files to implement blue slime's behavior.
- Goal: **The blue slime should alternate between moving in a random direction and stopping.**

TODO 2.1: Wanderer.ts

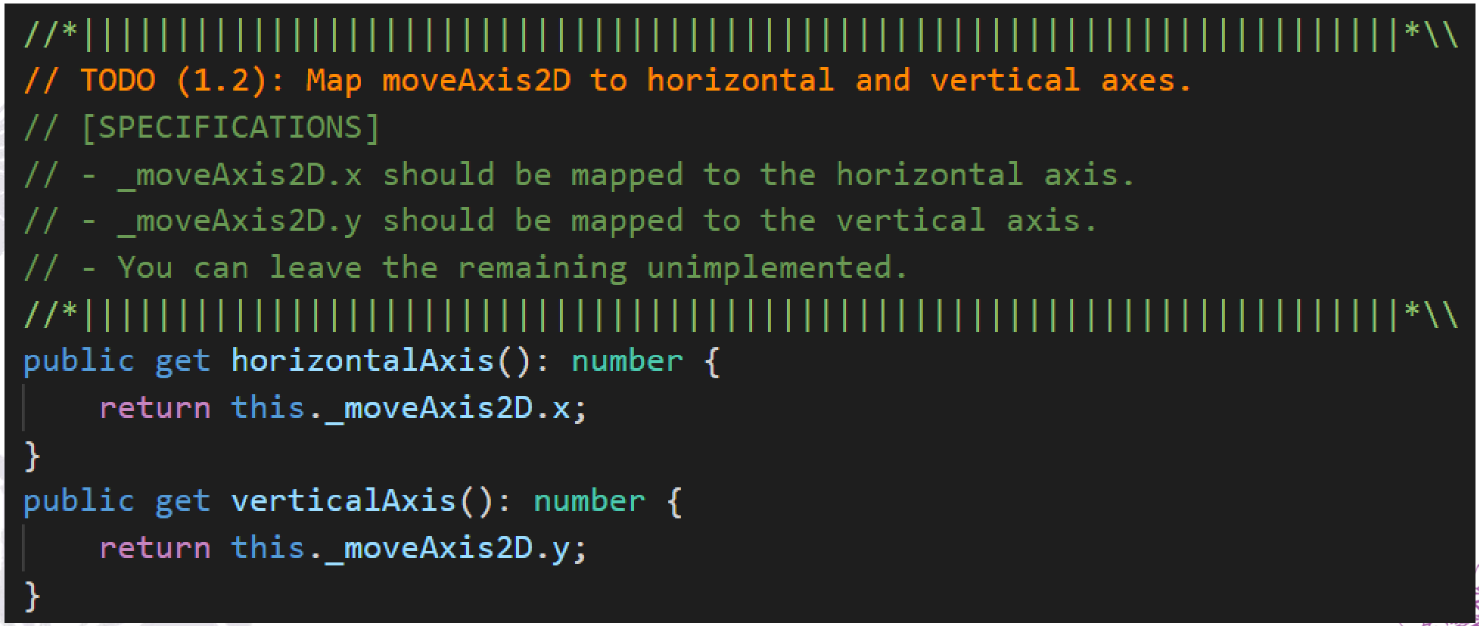
- Complete the constructor as follows:



```
/** The agent will move for this long before stopping to wait. */
private _moveDuration = 1.0;
/** The agent will wait for this long before starting to move again. */
private _waitDuration = 0.5;
/** The actual wait duration will be randomized by this factor,
 *  such that the actual wait duration is a random number between
 *  waitDuration x (1 - waitRandomFactor) and
 *  waitDuration x (1 + waitRandomFactor).
*/
 private _waitRandomFactor = 0.1;

constructor(moveDuration:number, waitDuration:number, waitRandomFactor:number) {
    super();
    //*|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||*\\
    // TODO (1.1): Complete the constructor.
    // [SPECIFICATIONS]
    // - Initialize the four private variables above properly.
    //*|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||*\\
    this._moveDuration = moveDuration;
    this._waitDuration = waitDuration;
    this._waitRandomFactor = waitRandomFactor;
}
```

TODO 2.2: Wanderer.ts

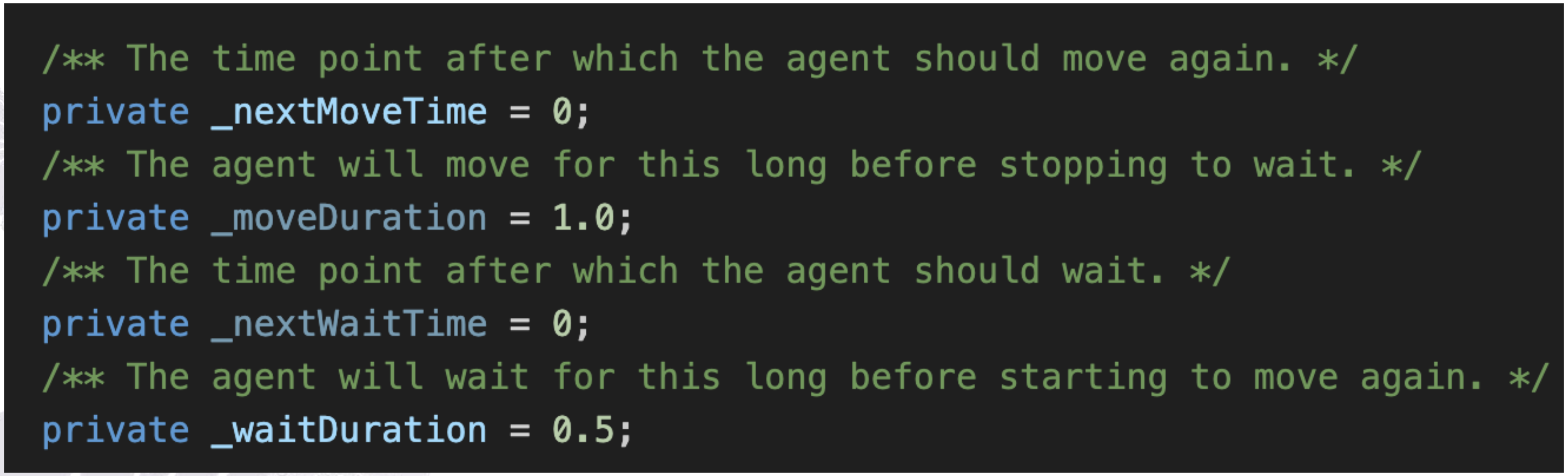
- Implement the **horizontalAxis**() and **verticalAxis**() methods with a Vec2 variable **_moveAxis2D**.



```
//*|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||*\\
// TODO (1.2): Map moveAxis2D to horizontal and vertical axes.
// [SPECIFICATIONS]
// - _moveAxis2D.x should be mapped to the horizontal axis.
// - _moveAxis2D.y should be mapped to the vertical axis.
// - You can leave the remaining unimplemented.
//*|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||*\\
public get horizontalAxis(): number {
    return this._moveAxis2D.x;
}
public get verticalAxis(): number {
    return this._moveAxis2D.y;
}
```

TODO 2.3: Wanderer.ts

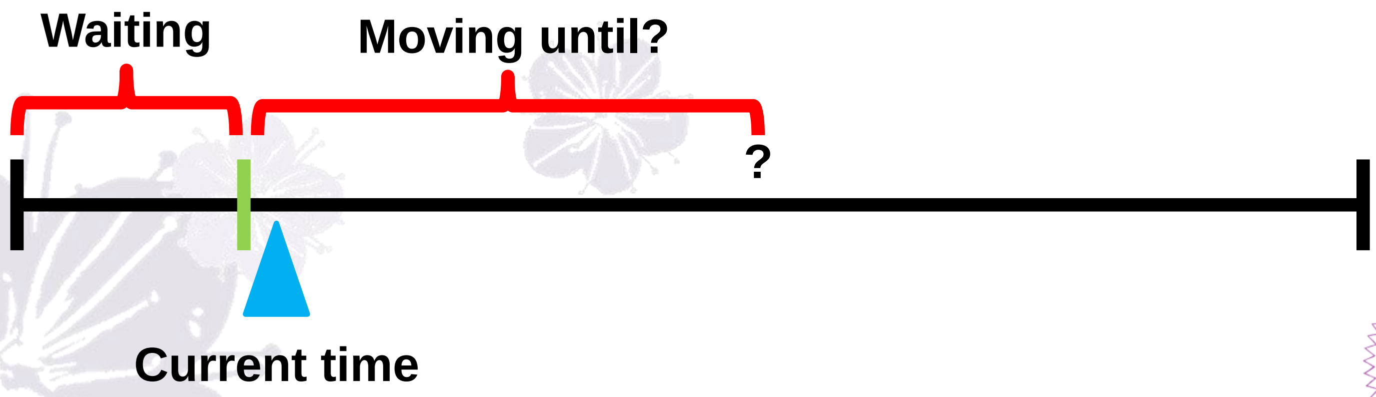
- Calculate and decide whether slime to move in the **update(dt)** function.
- Pay attention to four variables:



```
/** The time point after which the agent should move again. */
private _nextMoveTime = 0;
/** The agent will move for this long before stopping to wait. */
private _moveDuration = 1.0;
/** The time point after which the agent should wait. */
private _nextWaitTime = 0;
/** The agent will wait for this long before starting to move again. */
private _waitDuration = 0.5;
```

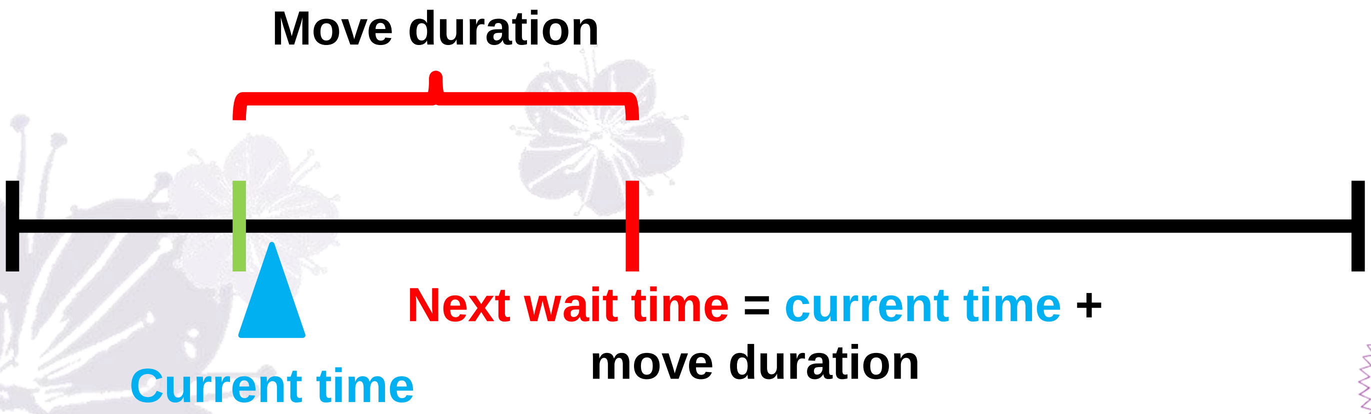
TODO 2.3: Wanderer.ts

- The agent should recompute its wandering direction when the current time (**currentTime**) reaches the next move time (**_nextMoveTime**).
  - Because of how time works, you can't use the equal operator here.



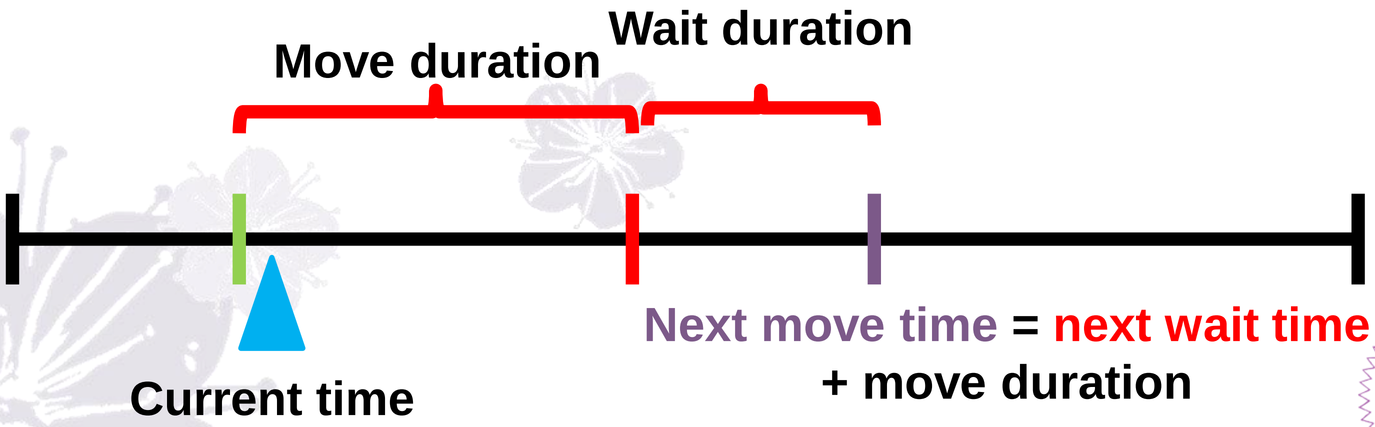
TODO 2.3: Wanderer.ts

- The next wait time (**_nextWaitTime**) is just the current time (**currentTime**) plus the move duration (**_moveDuration**), because the slime will move for (**_moveDuration**) seconds.



TODO 2.3: Wanderer.ts

- The next move time (**_nextMoveTime**) is then the next wait time pluses for how long the slime should wait (**_waitDuration**).

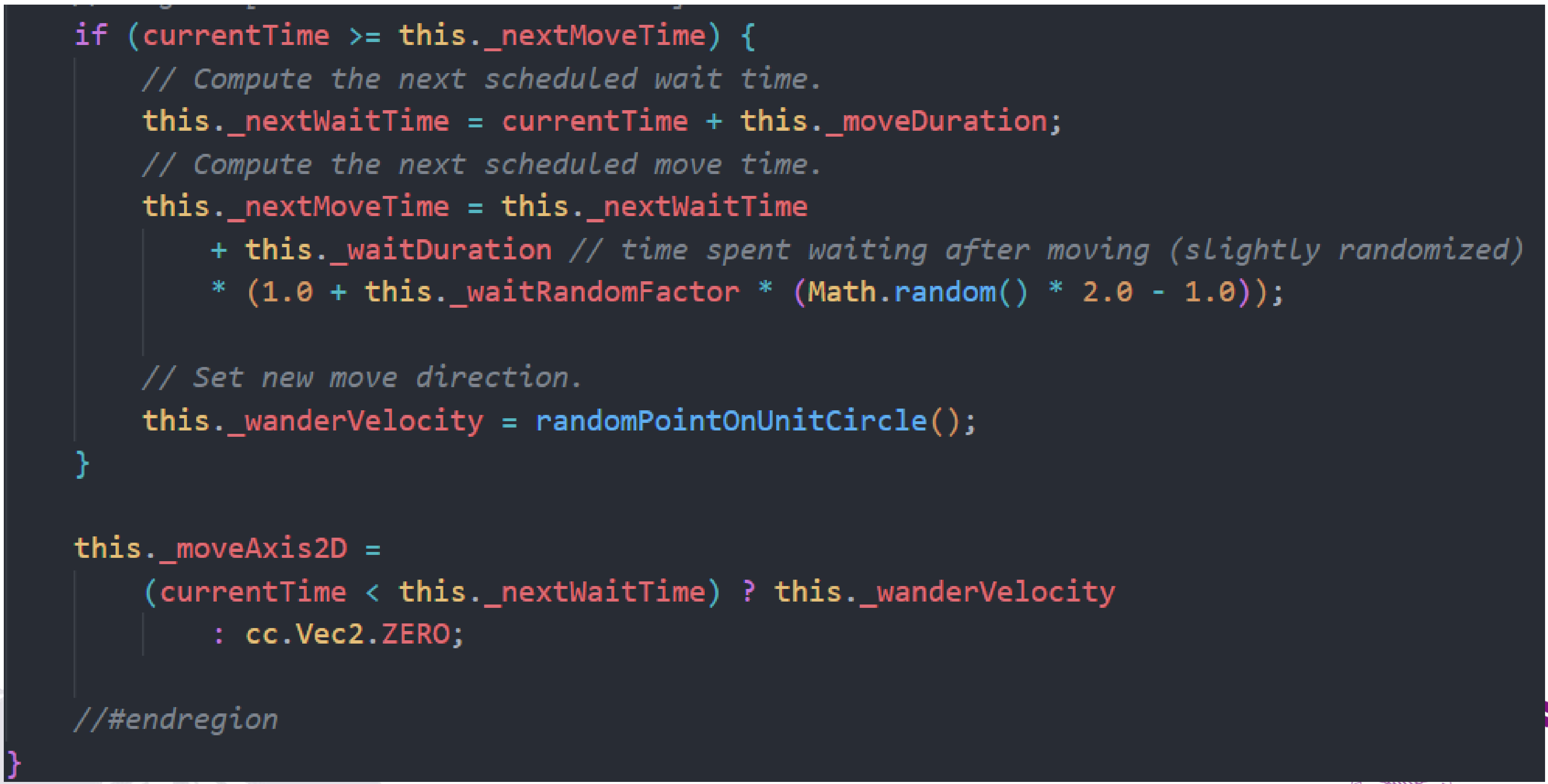


TODO 2.3: Wanderer.ts

- You can use the wait random factor (**_waitRandomFactor**) to randomize the wait duration. Details are given in the comments for _waitRandomFactor.
- If the agent reaches the next movement state, set the **_wanderVelocity** to **randomPointOnUnitCircle**().
- If the agent is in the moving state, set its 2D move axis **(_moveAxis2D**) to the **_wanderVelocity,** otherwise set to cc.Vec2.ZERO.

TODO 2.3: Wanderer.ts

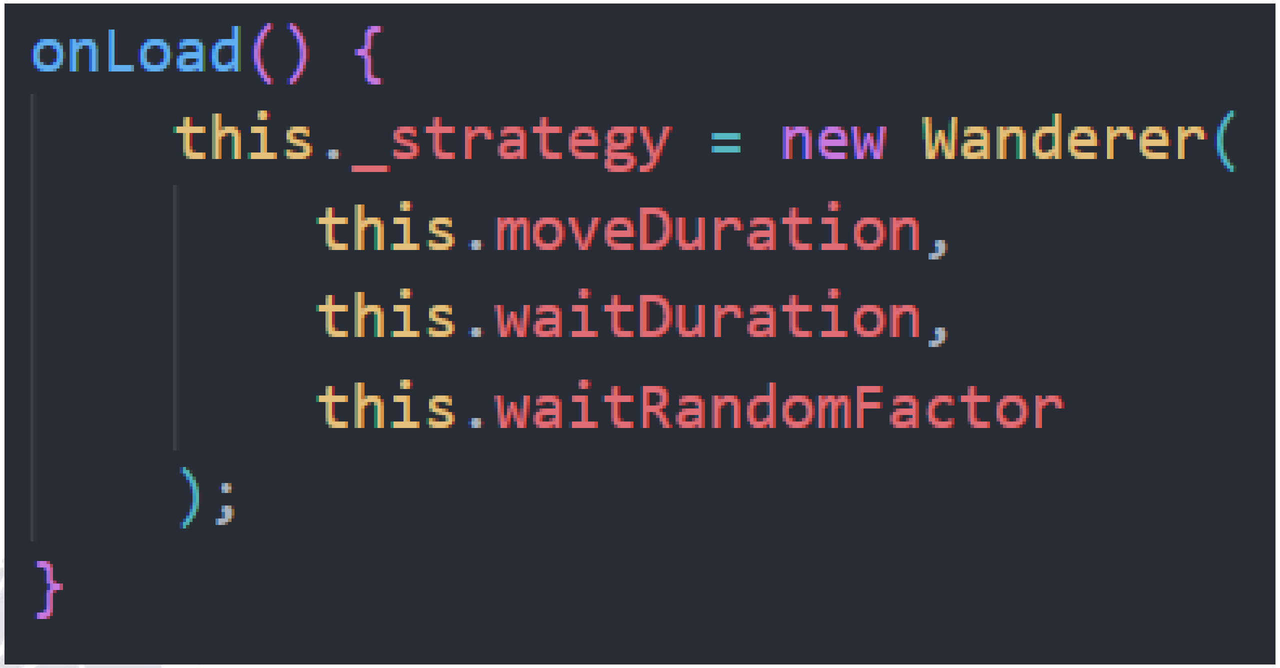
- Answer:



```
    if (currentTime >= this._nextMoveTime) {
        // Compute the next scheduled wait time.
        this._nextWaitTime = currentTime + this._moveDuration;
        // Compute the next scheduled move time.
        this._nextMoveTime = this._nextWaitTime
            + this._waitDuration // time spent waiting after moving (slightly randomized)
            * (1.0 + this._waitRandomFactor * (Math.random() * 2.0 - 1.0));

        // Set new move direction.
        this._wanderVelocity = randomPointOnUnitCircle();
    }

    this._moveAxis2D =
        (currentTime < this._nextWaitTime) ? this._wanderVelocity
            : cc.Vec2.ZERO;

    //#endregion
}
```

TODO 2.4: WanderAgent.ts

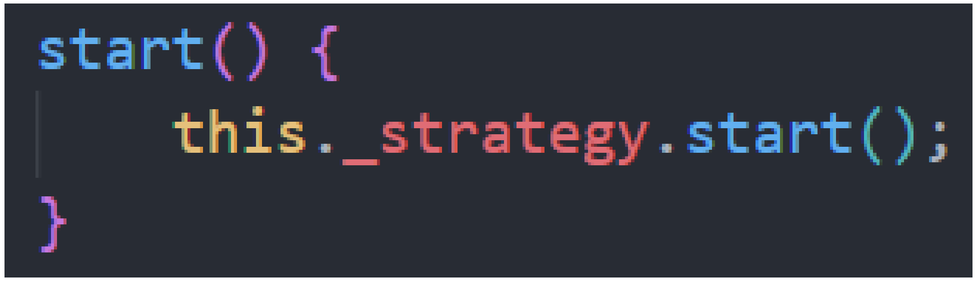
- Create a **Wanderer** strategy object in the Onload() function.



```
onLoad() {
    this._strategy = new Wanderer(
        this.moveDuration,
        this.waitDuration,
        this.waitRandomFactor
    );
}
```

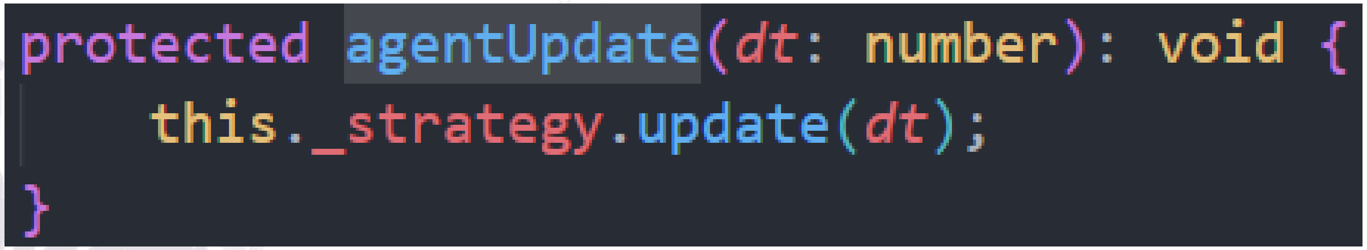
TODO 2.4: WanderAgent.ts

- Start the strategy in Start()



```
start() {
    this._strategy.start();
}
```

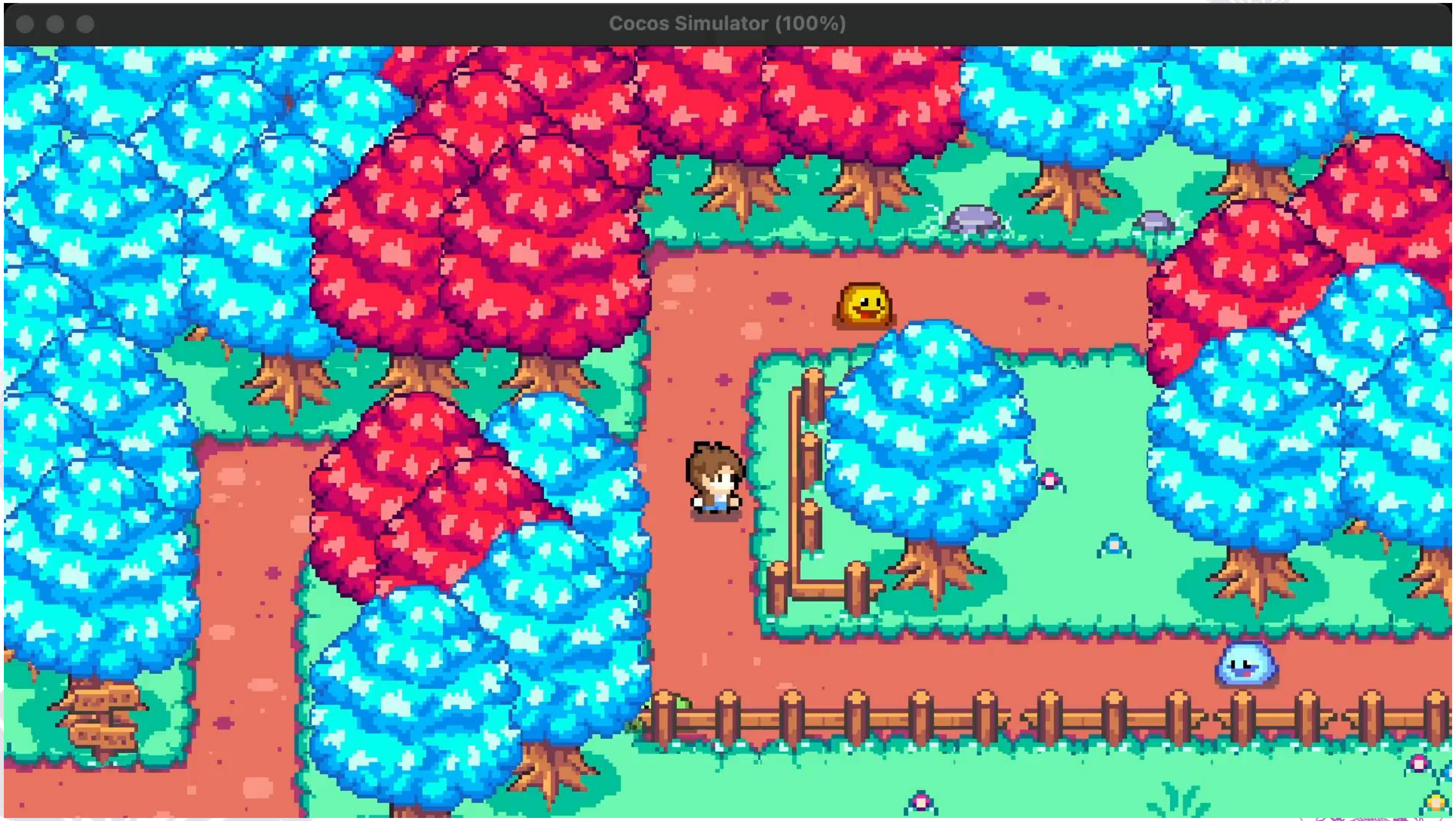
- Update the strategy in agentUpdate()

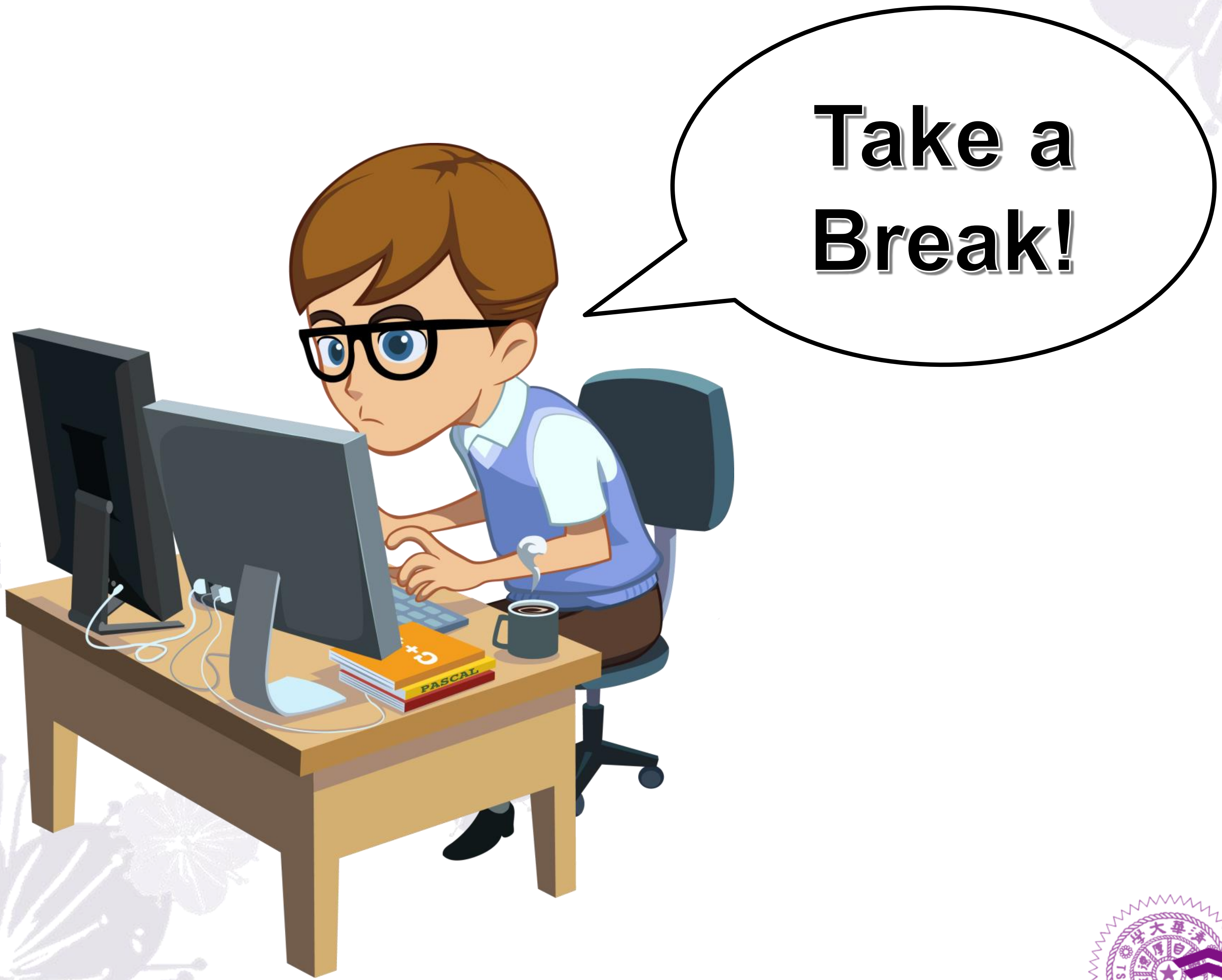


```
protected agentUpdate(dt: number): void {
    this._strategy.update(dt);
}
```

Note: The life-cycle method update(dt) is invoked in the parent Agent class! See Agent.ts.

Result



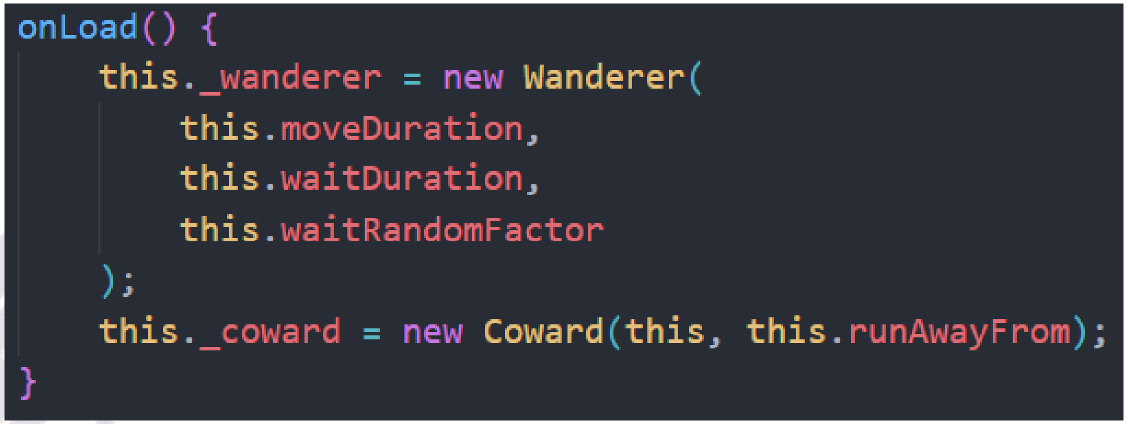

Take a Break!
C++
PASCAL

TODO 3

- File: **ShyAgent.ts (3.1, 3.2)**
- Follow the hints in the file to implement green slime's behavior.
- Goal: **The green slime should run away from the player when they get too close and go back to wandering once far away enough from the player.**

TODO 3.1: ShyAgent.ts

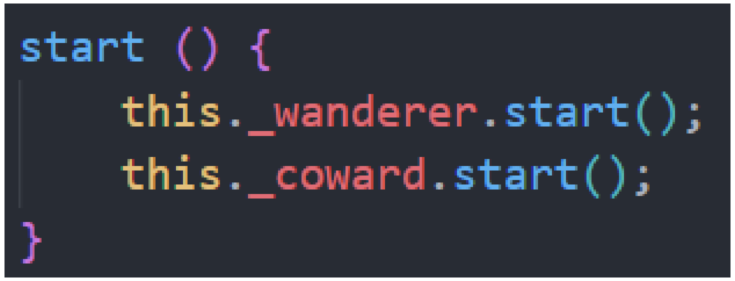
- Since this slime has two strategies, we first create two strategy objects for **_wanderer** and **_coward**.
- The Coward strategy needs to know the agent’s status: Pass **“this”** to it!



```
onLoad() {
    this._wanderer = new Wanderer(
        this.moveDuration,
        this.waitDuration,
        this.waitRandomFactor
    );
    this._coward = new Coward(this, this.runAwayFrom);
}
```

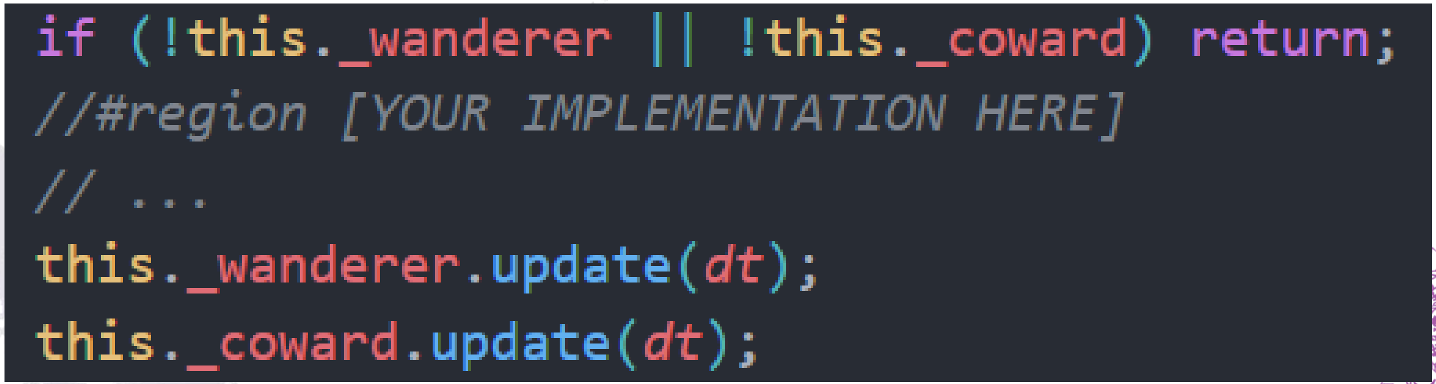
TODO 3.1: ShyAgent.ts

- Start two strategies



```
start () {
    this._wanderer.start();
    this._coward.start();
}
```

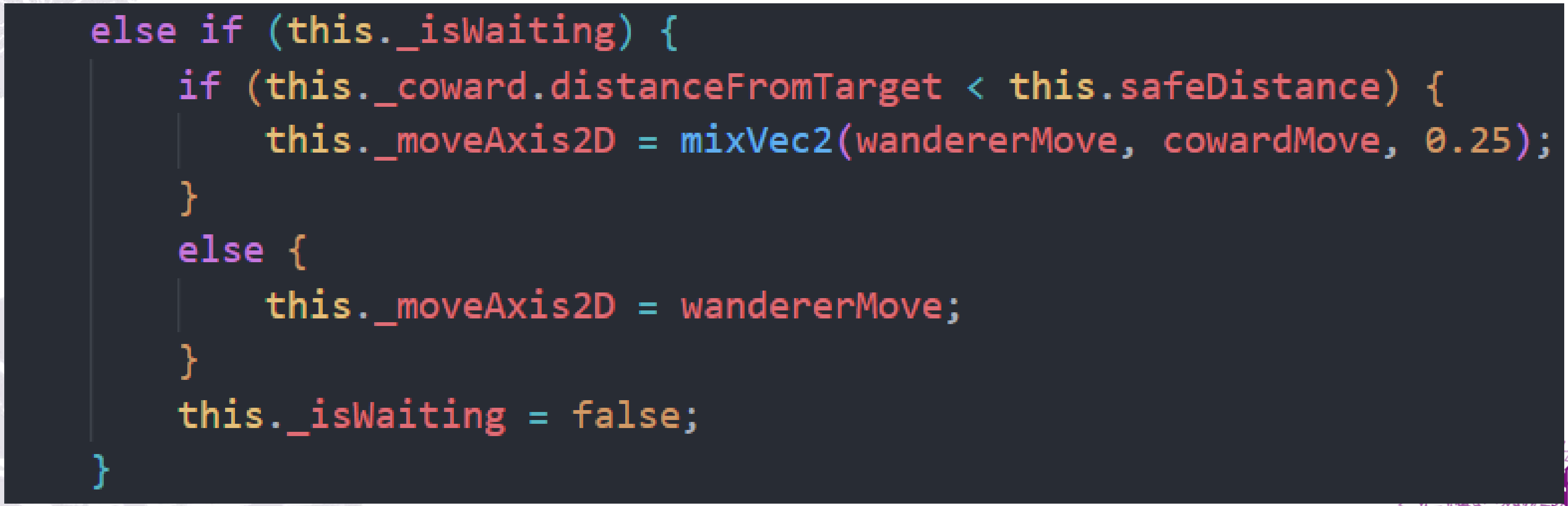
- Update two strategies in the **agentUpdate**()



```
if (!this._wanderer || !this._coward) return;
//#region [YOUR IMPLEMENTATION HERE]
// ...
this._wanderer.update(dt);
this._coward.update(dt);
```

TODO 3.2: ShyAgent.ts

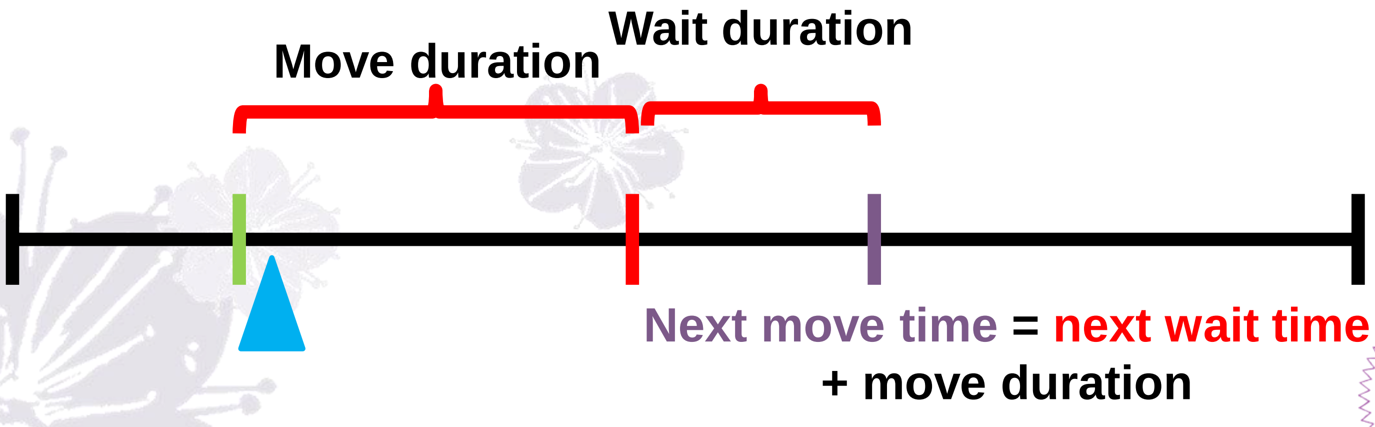
- “When it is about to move”
  - Equivalent to: When wandererMove isn’t equal to zero.
  - Write the following codes in the **agentUpdate**()



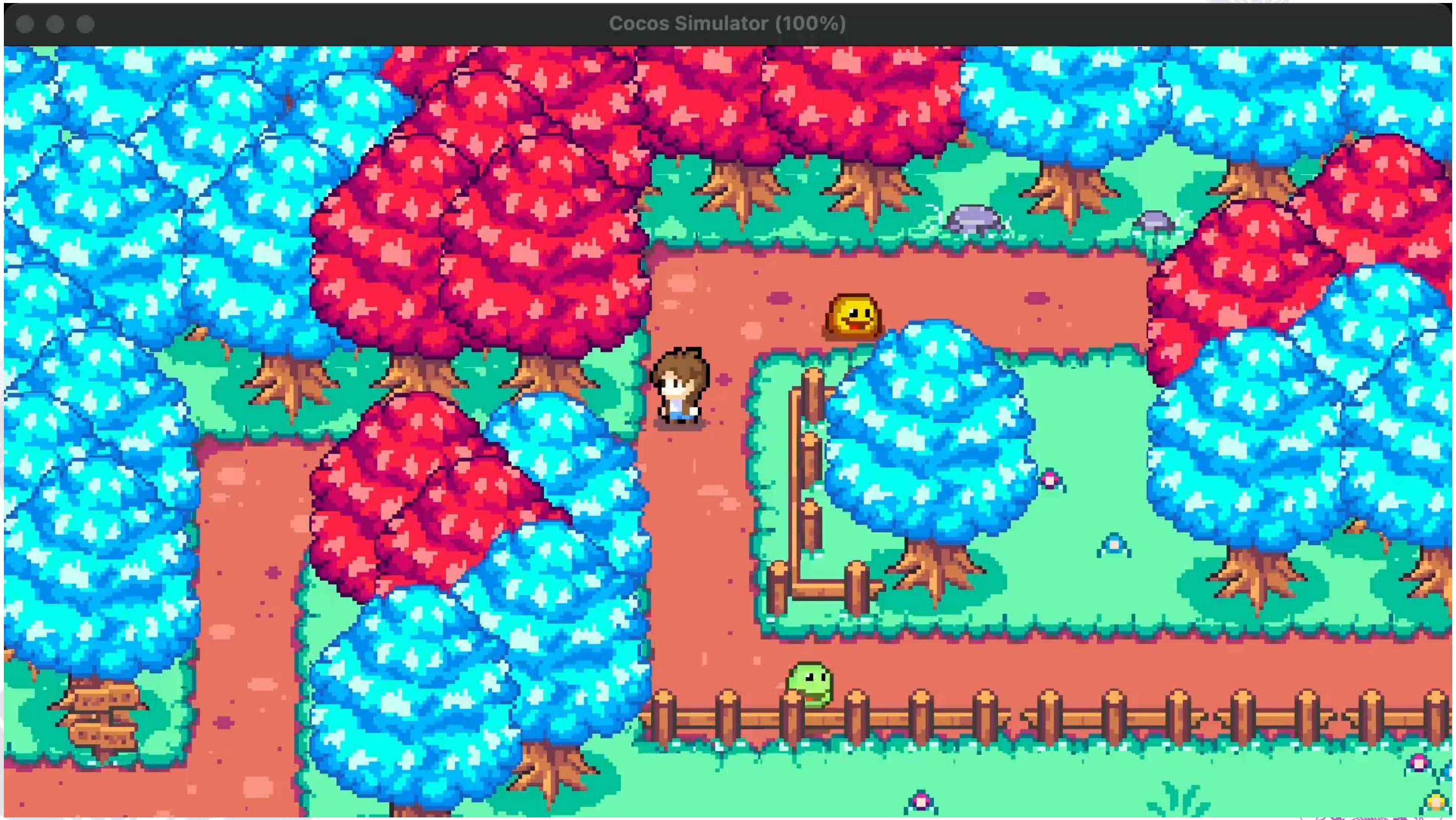
```
else if (this._isWaiting) {
    if (this._coward.distanceFromTarget < this.safeDistance) {
        this._moveAxis2D = mixVec2(wandererMove, cowardMove, 0.25);
    }
    else {
        this._moveAxis2D = wandererMove;
    }
    this._isWaiting = false;
}
```

TODO 3.2: ShyAgent.ts

- If wanderer enter the wait duration, _isWaiting will be set to true.
- In next frame, program will run into「else if(_isWaiting)」block, and set the moveAxis2D for next move time.



Result



thank you!

Question?