maple

unknown

Apr 05, 2023

CONTENTS

CHAPTER

ONE

MISSION

Maple is a project dedicated to creating an easier enviornment for new modders.

CHAPTER

TWO

DOCUMENTATION CONTENTS

2.1 Installation

Warning: This may change at any time as the project is still in development.

2.1.1 Installation of the Cinnamon Modloader

Prerequisites

- Python 3.9 (32-bit)

Installation

1. Copy `python39.dll` from the python directory to the Geometry Dash directory.
2. Download the latest release from the releases page.
3. Extract the zip file to a folder of your choice.
4. Load the `cinnamon.dll` with any dll loader.

Note: This process will be automated in the after the first full release.

2.2 Getting Started (for developers)

2.2.1 Your first mod (Python)

Setup

The first step in creating a mod is to create a new directory in `Geometry Dash/cinnamon/mods/`. The name of the directory will be the name of your mod.

Note: You may create a mod with a single file, but it is recommended to create a directory for your mod to keep things organized.

For this example we will create a mod called `button_test`, an example mod from the github repo. You can find more examples in the `examples` directory.

We will create a file named `button_test.py` in the `button_test` directory.

The code

The first thing we need to do is import the `cinnamon` module. This is the module that contains all the functions and classes that are used to create mods. We will also import the nessessary modules such as `cocos2d`, the game engine the game uses and `geometry_dash`, the functions from the game.

```
import cinnamon
import cocos2d
import geometry_dash
```

Next we need to create a class that inherits from `cinnamon.Mod`. This class will contain all the code for our mod.

```
class ButtonTest(cinnamon.Mod):
    pass
```

Now we need to create a function that will be called when the mod is loaded, the `__init__` function. This function will be called when the mod class is created.

We also need to call the `__init__` function of the `cinnamon.Mod` class to ensure that our mod is loaded correctly.

```
...

class ButtonTest(cinnamon.Mod):
    def __init__(self):
        cinnamon.Mod.__init__(self)
        prnt("Button Test mod loaded!")
```

One last thing before we can test our basic mod.

We need to register our mod with cinnamon. We do this by calling `cinnamon.register_mod` and passing our mod class as an argument.

```
...

class ButtonTest(cinnamon.Mod):
    def __init__(self):
        cinnamon.Mod.__init__(self)
        print("Button Test mod loaded!")

cinnamon.register_mod(ButtonTest)
```

Testing

Now try opening the game and see if your mod works. Debug versions of cinnamon automatically come with a console that you can use to print messages to.

If you see the message `Button Test mod loaded!` in the console then your mod is working!

You might notice that no buttons have been added to the game yet. We will add buttons in the next section.

Adding buttons

Now that we have a basic mod working we can add some buttons to the game. Before we can add buttons we need to create a new class that inherits from `geometry_dash.MenuLayer`. This class contains all of the code for the menu.

To add this button we need to **override** the `init` function of MenuLayer by **hooking** it.

This is how it is done in cinnamon.

```
...

class MenuLayer(geometry_dash.MenuLayer):
    @cinnamon.hook(geometry_dash.MenuLayer.init) # Hook the init function
    def init(self):
        self.init0() # we need to call the original function so the rest of the menu is␣
↪loaded

        # TODO: Add buttons here

class ButtonTest(cinnamon.Mod):
    def __init__(self):
        cinnamon.Mod.__init__(self)
        print("Button Test mod loaded!")

...
```

We can now add our button to the menu.

Our button needs to have a **sprite**, a **position** and a **callback**.'

A **callback** is the code that will run when the button is pressed.

For it to work, our button needs to be in a **CCMenu**, which is a cocos2d class that handles buttons and allows them to be clicked.

```
import cinnamon
import cocos2d
import geometry_dash

class MenuLayer(geometry_dash.MenuLayer):
    @cinnamon.hook(geometry_dash.MenuLayer.init) # Hook the init function
    def init(self):
        result = self.init0() # we need to call the original function so the rest of the␣
↪menu is loaded

        # Create a sprite for our button
        sprite = cocos2d.CCSprite.create("GJ_button_01.png")
```

(continues on next page)

(continued from previous page)

```
        # Create a button with our sprite

        button = geometry_dash.CCMenuItemSpriteExtra.create(sprite, self, self.button_
↪callback)

        # Create a CCMenu with our button

        menu = cocos2d.CCMenu.create(button)
        menu.addChild(button)

        # Add the menu to the layer

        self.addChild(menu, 99)

        director = cocos2d.CCDirector.sharedDirector() # Get the director
        win_size = director.getWinSize() # Get the size of the window

        menu.setPosition(win_size.width / 2, win_size.height / 2) # Set the position of␣
↪the menu to the center

        return result

    # this is the callback for our button
    def button_callback(self, sender):
        print("Button pressed!")

class ButtonTest(cinnamon.Mod):
    def __init__(self):
        cinnamon.Mod.__init__(self)
        print("Button Test mod loaded!")

cinnamon.register_mod(ButtonTest)
```

Final Test

Now try running the game again and see if your button works.

If you see the message `Button pressed!` in the console then your button is working! Congratulations, you have created your first mod!