

TÁC GIẢ: 8 SYNC DEV

MINECRAFT GAME PYTHON VERSION 1

Theo dõi kênh: [Tại đây](#)

Sẽ có cập nhật trong tương lai 😊 !!!

Video Hướng Dẫn Chi Tiết Phần 1: [Xem tại đây](#)

Video Hướng Dẫn Chi Tiết Phần 2 (Đa người chơi) Mới: [Xem tại đây](#)

Xem Tài Liệu Đa Người Chơi: [Xem tại đây](#)

Website: [Xem tại đây](#)

Tài liệu sẽ được cập nhật định kì và thông báo trong group nên các bạn chú ý nhen

Các khóa học khác xem tại : [8 Sync Dev](#)

Hoặc liên hệ trực tiếp qua fb: [Kevin Nguyễn](#) để được hỗ trợ

CÁU TRÚC DỰ ÁN



{ 8 Sync Dev }

```
src
├── settings
│   └── settings.py
├── resources
│   └── load_textures.py
└── entity
    └── event
        └── evt_pick.py
```

```
| |   └── evt_distance.py  
|   └── scene  
|       ├── block.py  
|       ├── sky.py  
|       └── hand.py  
|   └── genmap/  
└── character/  
└── main.py
```

Để minh họa trực quan, đây là cách tổ chức cấu trúc thư mục của dự án:

1. Thư mục gốc **src**:

- **Thư mục settings:**

- `settings.py`: Chứa các cài đặt và cấu hình cho dự án.

- **Thư mục resources:**

- `load_textures.py`: Tải các tài nguyên và kết cấu cần thiết cho trò chơi.

- **Thư mục entity:**

- **Thư mục event:**

- `evt_pick.py`: Xử lý sự kiện chọn.
 - `evt_distance.py`: Xử lý sự kiện khoảng cách.

- **Thư mục scene:**

- `block.py`: Định nghĩa khối trong trò chơi.
 - `sky.py`: Định nghĩa bầu trời trong trò chơi.
 - `hand.py`: Định nghĩa bàn tay trong trò chơi.
 - `genmap.py`: Tạo bản đồ trong trò chơi.
 - `character.py`: Định nghĩa nhân vật trong trò chơi.

- `main.py`: Tệp chính để chạy ứng dụng.

CHI TIẾT VỀ CÁC TỆP
TỆP SETTINGS.PY



{ 8 Sync Dev }

```
from pathlib import Path

BASE_DIR = Path(__file__).resolve().parent.parent.parent

MODEL_PATH = str(BASE_DIR / 'assets' / 'models')
SFX_PATH = str(BASE_DIR / 'assets' / 'sfx')
TEXTURE_PATH = str(BASE_DIR / 'assets' / 'textures')
```

TÊP LOAD_TEXTURES.PY



{ 8 Sync Dev }

```
from ursina import load_texture, Audio
from src.settings import TEXTURE_PATH, SFX_PATH

# Load textures
grass_texture = load_texture(f'{TEXTURE_PATH}\\Grass_Block.png')
stone_texture = load_texture(f'{TEXTURE_PATH}\\Stone_Block.png')
brick_texture = load_texture(f'{TEXTURE_PATH}\\Brick_Block.png')
dirt_texture = load_texture(f'{TEXTURE_PATH}\\Dirt_Block.png')
wood_texture = load_texture(f'{TEXTURE_PATH}\\Wood_Block.png')
sky_texture = load_texture(f'{TEXTURE_PATH}\\Skybox.png')
arm_texture = load_texture(f'{TEXTURE_PATH}\\Arm_Texture.png')

# Load sounds
punch_sound = Audio(f'{SFX_PATH}\\Punch_sound.wav', loop=False,
autoplay=False)
```

TÊP EVT_PICK.PY



{ 8 Sync Dev }

```
from ursina import held_keys
from src.entity.scene import hand, Block

def evt_pick():
    if held_keys['left mouse'] or held_keys['right mouse']:
        hand.active()
    else:
        hand.passive()

    if held_keys['1']: Block.block_pick = 1
    if held_keys['2']: Block.block_pick = 2
    if held_keys['3']: Block.block_pick = 3
    if held_keys['4']: Block.block_pick = 4
    if held_keys['5']: Block.block_pick = 5
```



{ 8 Sync Dev }



```
from src.entity.genmap import voxel, noise
from src.entity.character import character
from src.entity.scene import Block

distance = 10

def evt_distance():
    player_x = int(character.position.x)
    player_z = int(character.position.z)

    for z in range(player_z - distance, player_z + distance):
        for x in range(player_x - distance, player_x + distance):
```

```
if (x, z) not in voxel:  
    y = int(noise([x * 0.1, z * 0.1]) * 10)  
    block = Block(position=(x, y, z))  
    voxel[(x, z)] = block
```

TẾP GENMAP.PY

{ 8 Sync Dev }

```
from perlin_noise import PerlinNoise  
from src.entity.scene import Block  
import random  
  
noise = PerlinNoise(octaves=3, seed=random.randint(0,  
1000000000))  
voxel = {}  
  
def gen_map():  
    for z in range(20):  
        for x in range(20):  
            y = int(noise([x * 0.1, z * 0.1]) * 10)  
            block = Block(position=(x, y, z))  
            voxel[(x, z)] = block
```

TẾP BLOCK.PY

{ 8 Sync Dev }

```
from ursina import Button, destroy, scene, color, mouse  
import random  
from src.resources import grass_texture, stone_texture,  
brick_texture, dirt_texture, wood_texture, punch_sound
```

```
from src.settings import MODEL_PATH

class Block(Button):
    block_pick = 1

    def __init__(self, position=(0, 0, 0),
texture=grass_texture):
        super().__init__(
            parent=scene,
            position=position,
            model=f'{MODEL_PATH}\\Block',
            origin_y=0.5,
            texture=texture,
            color=color.color(0, 0, random.uniform(0.9, 1.0)),
            scale=0.5
        )

    def input(self, key):
        if self.hovered:
            if key == 'left mouse down':
                punch_sound.play()
                destroy(self)
            elif key == 'right mouse down':
                punch_sound.play()
                if Block.block_pick == 1:
                    voxel = Block(position=self.position +
mouse.normal, texture=grass_texture)
                elif Block.block_pick == 2:
                    voxel = Block(position=self.position +
mouse.normal, texture=stone_texture)
                elif Block.block_pick == 3:
                    voxel = Block(position=self.position +
mouse.normal, texture=brick_texture)
                elif Block.block_pick == 4:
                    voxel = Block(position=self.position +
mouse.normal, texture=dirt_texture)
                elif Block.block_pick == 5:
```

```
        voxel = Block(position=self.position +  
mouse.normal, texture=wood_texture)
```

TẾP SKY.PY

```
from ursina import Entity, scene  
from src.resources import sky_texture  
  
class Sky(Entity):  
    def __init__(self):  
        super().__init__()  
        parent=scene,  
        model='Sphere',  
        texture=sky_texture,  
        scale=150,  
        double_sided=True  
  
sky = Sky()
```

TẾP HAND.PY

```
from ursina import Entity, camera, Vec3, Vec2  
from src.resources import arm_texture  
from src.settings import MODEL_PATH  
  
class Hand(Entity):  
    def __init__(self):
```

```
self.passive_pos = Vec2(0.3, -0.6)
self.active_pos = Vec2(0.3, -0.5)
super().__init__(
    parent=camera.ui,
    model=f'{MODEL_PATH}\\Arm',
    texture=arm_texture,
    scale=0.2,
    rotation=Vec3(150, -10, 0),
    position=self.passive_pos
)

def active(self):
    self.position = self.active_pos

def passive(self):
    self.position = self.passive_pos

hand = Hand()
```

TẾP CHARACTER.PY

```
from ursina.prefabs.first_person_controller import
FirstPersonController

character = FirstPersonController()
```

TẾP MAIN.PY

```
{ 8 Sync Dev }
```

```
from ursina import Ursina
from src.resources import * # Tải tất cả các tài nguyên
from src.entity.scene import sky, hand, Block
from src.entity.genmap import gen_map
from src.entity.character import character
from src.entity.event import update

app = Ursina()

gen_map()
app.run()
```

CỘNG ĐỒNG

KHÓA HỌC:



[Kevin Nguyễn](#)



[Fullstack Python](#)



[Nhóm Chia Sẻ Công Nghệ](#)



[Fullstack Nextjs](#)



[Nhóm BlockChain](#)



[Fullstack Android-IOS](#)



[Tiktok: 8 Sync](#)



[Youtube: 8 Sync Dev](#)



[Zalo](#)

Tài liệu sẽ được cập nhật định kì và thông báo trong group nên các bạn chú ý nhen .

Các khóa học khác xem tại : [8 Sync Dev](#)

Hoặc liên hệ trực tiếp qua fb: [Kevin Nguyễn](#) để được hỗ trợ

