Underwater treasure hunt game

Python Programming Final Project

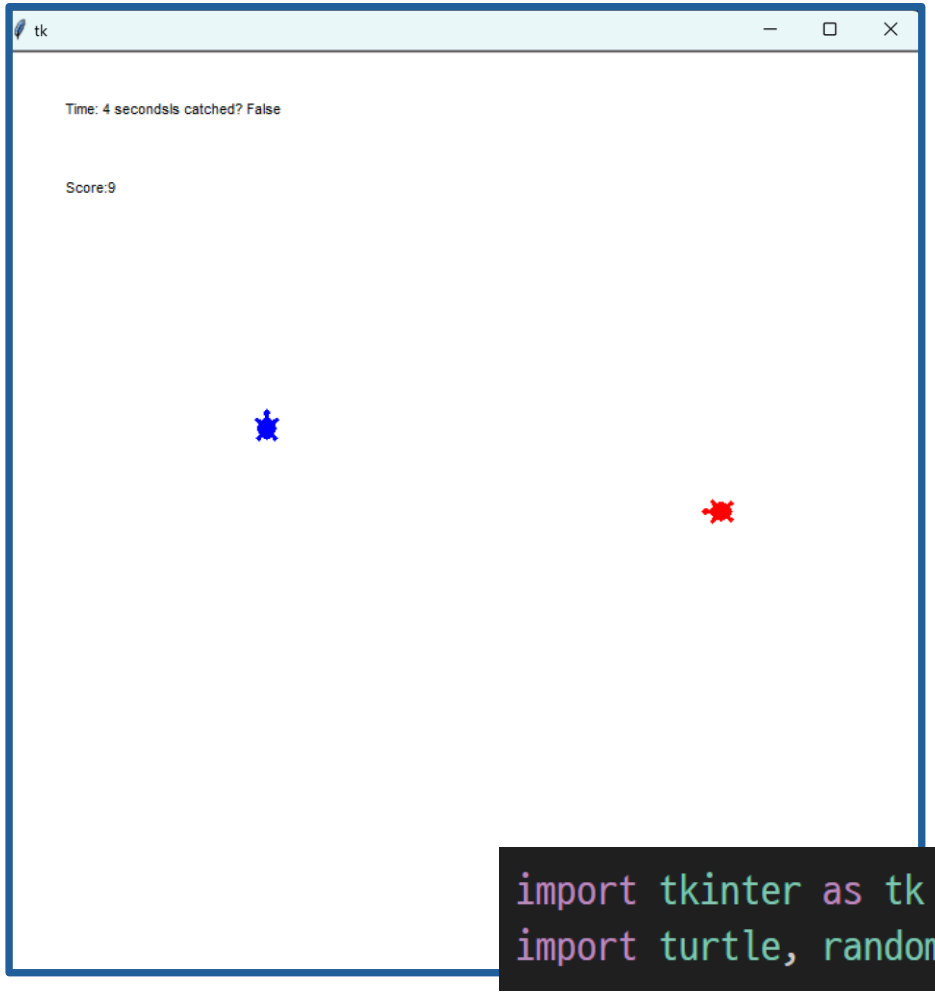
22102326 Yang Soyeon
23101943 Lee Jihyang

Contents

1. Purpose

Why did we do this project?

Purpose




```
import tkinter as tk
import turtle, random, time
```

- We have previous experience using **Turtle graphics** to implement games.
- We wanted to add more contents, and to implement it in **3D**.
- In the meantime, We came across a Python library called **ursina**

Appendix. What is Ursina?

Ursina is a 3D game engine written in Python.
We can easily make 3D games using it.

The advantages of Ursina

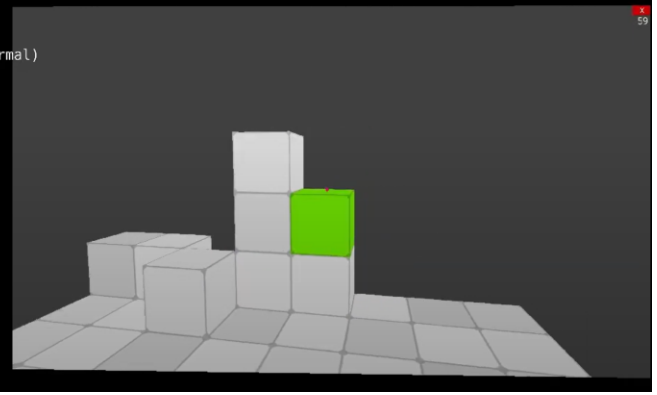
- Open source

- Can easily define objects using Entity class (location, model and texture, behavior, etc.)

- Basic UI elements such as buttons are built in, making it easy to add interfaces.

- It supports animation.

```python
from ursina import *
from ursina.prefabs.first_person_controller import FirstPersonController

app = Ursina()

class Voxel(Button):
    def __init__(self, position=(0,0,0)):
        super().__init__(
            parent = scene,
            position = position,
            model = 'cube',
            origin_y = .5,
            texture = 'white_cube',
            color = color.color(0, 0, random.uniform(.9, 1.0)),
            highlight_color = color.lime,
        )

    def input(self, key):
        if self.hovered:
            if key == 'left mouse down':
                voxel = Voxel(position=self.position + mouse.normal)

            if key == 'right mouse down':
                destroy(self)

for z in range(8):
    for x in range(8):
        voxel = Voxel(position=(x,0,z))

player = FirstPersonController()
app.run()
```

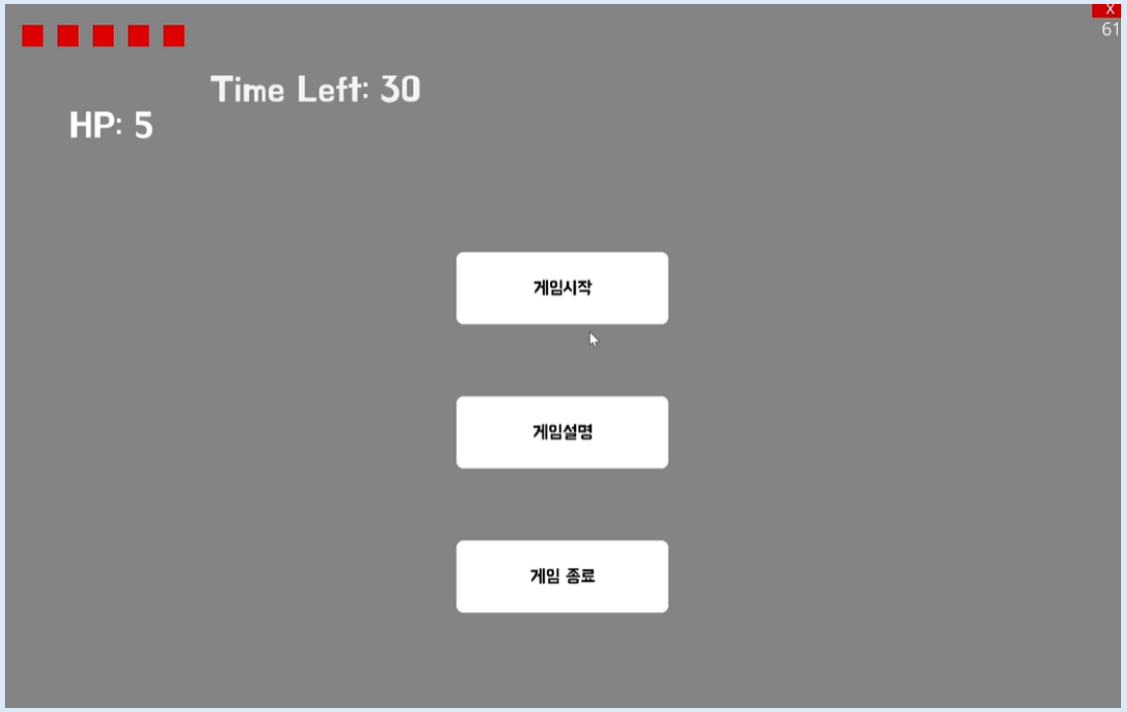


2. Core Structure

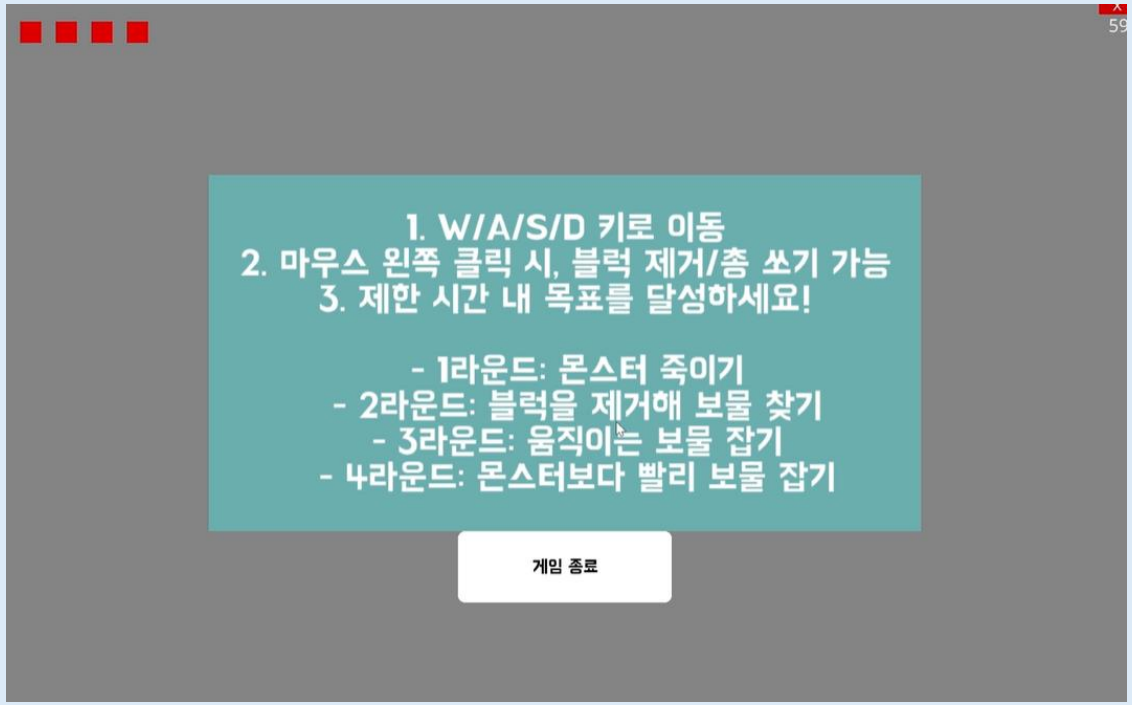
What functions does our program have?

Core Structure

Main Menu



Click '게임설명' button



Core Structure

- Click ‘게임시작’ button

Round 0

You need to shoot and <u>kill monsters</u> that are approaching you. The gun can be shot with a <u>left click</u> of the mouse.

Round 1

You need to <u>find hidden treasures</u> by removing blocks that have been created. You can remove blocks by <u>left-clicking</u> the mouse.

Core Structure

- Click ‘게임시작’ button

Round 2

You should click on the <u>moving treasure</u> appropriately.

Round 3

<u>Faster than a moving monster</u>, you have to click on a moving treasure. When the monster finds the treasure first, the game ends.

3. Algorithm

How did we solve the problem?

Algorithm

1. Configuring the screen

```python
def setup_main_menu(round):
    """메인 메뉴 화면 생성"""
    global menu_entities
```

```python
...
start_button = Button(
    parent=camera.ui,
    text="게임시작",
    color=color.white,
    text_color=color.black,
    scale=(0.3, 0.1),
    position=(0, 0.1),
    on_click=lambda: set_game_state('game'),
    z=-2
)
```

Sets the function to be executed when a button is clicked.

- Start game
- Quit game

Algorithm

2. FPS

```python
class CustomFPSController(Entity):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.speed = 5  # 이동 속도
        self.rotation_speed = 40  # 회전 속도
        self.camera_pivot = Entity(parent=self, y=1.5)  # 카메라 회전 축
        camera.parent = self.camera_pivot
        camera.position = (0, 1.5, 0)
        camera.rotation = (0, 0, 0)
        mouse.locked = False
```

```python
def update(self):
    """플레이어 업데이트: 이동 및 회전"""
    if not self.enabled:
        return  # 비활성화 상태에서는 아무 동작도 하지 않음

    self.rotation_y += mouse.velocity[0] * self.rotation_speed
    self.camera_pivot.rotation_x -= mouse.velocity[1] * self.rotation_speed
    self.camera_pivot.rotation_x = clamp(self.camera_pivot.rotation_x, -90, 90)

    move = Vec3(
        self.forward * (held_keys['w'] - held_keys['s']) +
        self.right * (held_keys['d'] - held_keys['a'])
    ).normalized() * self.speed * time.dt
    self.position += move
```

Algorithm

2. Creating a Player

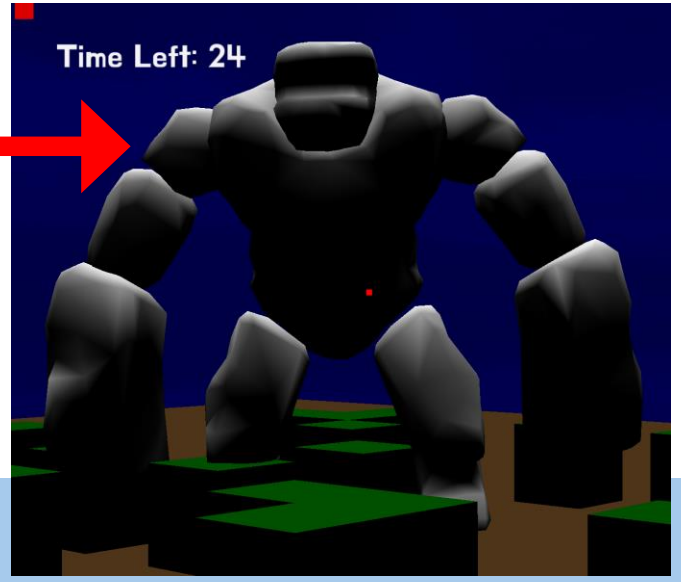
```python
class Player(CustomFPSController):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.hp_icons = []  # HP 아이콘 리스트
        self.max_hp = 5  # 최대 HP
        self.hp = self.max_hp
```

```python
def take_damage(self):
    """플레이어가 데미지를 받는 메서드"""
    if self.hp > 0:
        self.hp -= 1
        destroy(self.hp_icons.pop())  # HP 아이콘 제거
        self.hp_text.text = f"HP: {self.hp}"
        print(f"Player HP: {self.hp}")
        if self.hp <= 0:
            end_game("You were defeated by the monster!")
```

Algorithm

2. Creating a Monster

```python
class Monster(Entity):
    def __init__(self, target=None, **kwargs):
        super().__init__(
            model='stone_monster/Stone(28).obj',
            scale=(0.5, 0.5, 0.5),
            collider='box',
            **kwargs
```




```python
def apply_damage_to_player(self):
    """1초마다 플레이어에게 데미지를 가함"""
    if self.destroyed or not self.enabled:  # 삭제된 경우 종료
        self.attacking = False
        return

    if distance(self.position, self.target.position) < self.attack_distance:
        self.target.take_damage()  # 플레이어 HP 감소
        if self.target.hp > 0:
            invoke(self.apply_damage_to_player, delay=1)  # 1초 후 다시 실행
    else:
        self.attacking = False  # 플레이어와의 거리가 멀어지면 공격 중단

def take_damage(self, damage):
    """몬스터가 피해를 입음"""
    if self.destroyed:
        return
```

Created own 3D model using **blender**

Algorithm

2. Creating a Treasure

```python
class Treasure(Entity):
    def __init__(self, move=False, **kwargs):
        super().__init__(
            model='cube',
            texture='gold.jpg',
            #color=color.gold,
            scale=(1, 1, 1),
            collider='box',
            **kwargs
```

```python
if self.move:  # 보물이 움직이는 경우
    self.position += self.direction * time.dt * self.speed

    # 맵 경계 제한: 보물이 맵을 벗어나지 않도록 조정
    if abs(self.position.x) > 30 or abs(self.position.z) > 30:
        self.direction *= -1  # 방향 반전

    # 벽돌 근처에서만 움직이도록 제한
    nearest_brick = min(brick_positions, key=lambda pos: distance(Vec3(pos), self.position))
    if distance(Vec3(nearest_brick), self.position) > 5:  # 벽돌에서 일정 거리 이내로 제한
        self.direction *= -1
```

For treasure objects, **add moving properties** as well.

Algorithm

3. Implementation for each round

```
# 라운드에 따라 설정
if current_round == 0:
    # 라운드 0: 괴물이 플레이어를 추적
    monster = Monster(target=player, position=random.choice(brick_positions), enabled=True)
    monster.active = True
elif current_round == 1:
    # 라운드 1: 보물이 고정된 상태
    treasure = Treasure(position=random.choice(brick_positions), move=False)
elif current_round == 2:
    # 라운드 2: 보물이 움직이는 상태
    treasure = Treasure(position=random.choice(brick_positions), move=True)
elif current_round == 3:
    # 라운드 3: 괴물이 보물을 추적
    treasure_pos = random.choice(brick_positions)
    treasure = Treasure(position=treasure_pos, move=True)

    # 몬스터 등장
    possible_positions = [pos for pos in brick_positions if distance(pos, treasure_pos) > 10]
    monster_pos = random.choice(possible_positions) if possible_positions else Vec3(0, 1, 0)
    monster = Monster(target=treasure, position=monster_pos, enabled=True)
    monster.active = True
    print("Monster has spawned and is targeting the treasure!")
```

Algorithm

4. Map Design

Game title: Seek a treasure box chest away from alien life under the sea

The surrounding environment was dark blue and the floor was dark brown, indicating the seabed.

Alien life chose a huge brick character so that the player could feel the fear.

Treasure box was made to look full of money and green boxes were made to look like underwater coral reefs.

4. Demo Video

https://youtu.be/pcVAVKuBXPY

Thank you