



Универзитет „Св. Кирил и Методиј“ во Скопје  
**ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И  
КОМПЈУТЕРСКО ИНЖЕНЕРСТВО**

-Video Game Programming-  
**Laboratory Exercise 2: Space Scavenger**

Student: Blerona Muladauti      ID: 221541

Course Professor: Katarina Trojacanec

Course Assistant: Slave Temkov

December, 2024

# Table of Contents

1. Introduction.....	3
2. How to Play.....	3
3. Controls.....	4
4. Game Features.....	5
Dynamic Background: .....	5
Power-ups:.....	5
Scaling Difficulty: .....	5
Sound Effects and Music: .....	5
Game Over Mechanism: .....	5
Visual Effects: .....	5
5. Implementation Details.....	6
5.1 Game Environment .....	6
5.2 Player Movement and Shooting.....	6
5.3 Dynamic Background.....	6
5.4 Obstacles and Power-ups.....	6
5.5 Collision Detection .....	6
5.6 Sound and Music .....	6
5.7 Power-ups and Effects.....	6
5.8 Game Over and Restart .....	6
6. Conclusion .....	9

# Introduction

"Space Scavenger" is a 2D space-themed arcade game where players navigate a spaceship to collect crystals, avoid asteroids, and shoot enemies. Player must use their skills to survive and earn the highest score possible, and each game try and beat their best score.

In Space Scavenger, survival is key. The game combines fast-paced action with intricate gameplay mechanics, requiring players to control the spaceship and shoot down asteroids with lasers while simultaneously dodging obstacles.



---

## How to Play

The primary goal is to collect crystals to earn points and power-ups while avoiding or destroying asteroids. The game ends when the player's lives reach zero. High scores are tracked during gameplay, and power-ups enhance the player's abilities.

Control the spaceship using the arrow keys, and spacebar key to shoot. By default player has 1 life, but they add by earning a certain score that is stated in the following section. The game lasts until player loses (0 lives left). Player loses lives by collision with the asteroids which he must avoid. One of the objectives of the game also includes collecting the powerup crystals. Initially spaceship can only move right and left, but with a certain earning as bonus the up and down will be unlocked.

## Game Objectives:

1. Collect Crystals: Increase score and unlock power-ups.
2. Avoid Asteroids: Dodge or destroy obstacles to survive.
3. Survive Waves: Progress through increasing difficulties over time.
4. Achieve High Scores: Compete for the highest score.

## Game Bonuses:




1. Each 5 crystals, spaceship gets a powerup for 5 seconds (this powerup includes laser shooting on all sides).
2. On the 300<sup>th</sup> earned score, vertical movement is unlocked.
3. Each 500 earned scores player earns an extra life.

Since the game ends when player loses, with time it gets more and more impossible. This is achieved by increasing the size and frequency of the asteroids which initially are rare and smaller. But in fairness, the spaceship also accelerates with time upping the chances of survival.

After game ends (player loses), all values are set back to default including the score, lives, crystals earned, spaceship speed, and asteroid size and frequency, only the high score is kept track of in every session.

## Controls

Space Scavenger has a simple game controlling, it only includes:

- **Arrow Keys (Left/Right):** Move the spaceship horizontally. 
- **Arrow Keys (Up/Down):** Move vertically (unlocked after reaching 300 points). 
- **Spacebar:** Shoot bullets to destroy asteroids. 
- **Escape:** Quit the game.

# Game Features

## Dynamic Background:

- A scrolling starfield provides a visual effect of movement.

## Power-ups:

- Crystals grant score bonuses and power-ups like multi-shot lasers.



Earned 5 crystals  
got powerup:

## Scaling Difficulty:

- Asteroids increase in size and frequency as gameplay progresses.

## Sound Effects and Music:

- Audio effects for background, laser shooting, collisions, and power-ups.

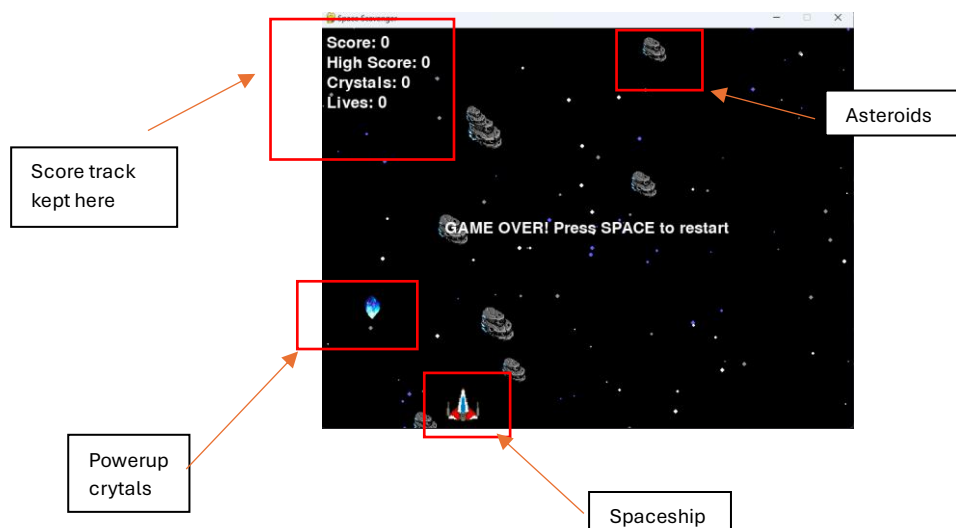
## Game Over Mechanism:

- Restart option available which sets all values to default and keeps track of high score.



## Visual Effects:

- Respawn with fading effects when spaceship has more than 1 life.



# Implementation Details

## 5.1 Game Environment

- **Window Setup:**
  - The game runs at 800x600 resolution with a 60 FPS frame rate.
  - Colors and fonts are defined globally for consistency.

## 5.2 Player Movement and Shooting

- Movement is controlled by key events and boundary checks.
- Shooting uses laser objects, which are updated with physics calculations.

## 5.3 Dynamic Background

- The scrolling star space view uses randomly positioned stars that continuously move downward, wrapping around the screen.

## 5.4 Obstacles and Power-ups

- **Asteroids:** Scale dynamically in size and frequency based on game time.
- **Crystals:** Provide score boosts and activate power-ups like multi-shot bullets.

## 5.5 Collision Detection

- Implemented using Pygame's built-in sprite collision detection to handle interactions between the player, bullets, asteroids, and crystals.

## 5.6 Sound and Music

- Background music and sound effects enhance immersion.
- Volume settings are pre-configured but can be adjusted.

## 5.7 Power-ups and Effects

- Multi-shot activation provides a spread pattern of bullets for rapid firing.
- Fading effects during respawn create visual invincibility feedback.

## 5.8 Game Over and Restart

- Players can restart after a game-over event without exiting the game.
- High scores persist until the game is closed.

# How things work in code

## 1. Power-Ups

- **Collected Crystals:**

```
for crystal in crystal_hits:
    crystal_sound.play()
    score += 10
    crystal_count += 1
    high_score = max(high_score, score)

    # Power-up activation every 5 crystals coll
    if crystal_count % 5 == 0:
        player.activate_power_up()
        active_messages.append(show_message(scr
```

self.power\_up\_active = True

self.power\_up\_timer = self.power\_up\_duration

## 2. Lives

- **Initial Lives:**

MAX\_LIVES = 3

*Starts with 1 life, can have up to 3 lives*

lives = 1

- **Extra Lives:**

```
if score % 500 == 0 and lives < MAX_LIVES:
    lives += 1
    active_messages.append(show_message(screen,
```

*Gain an extra life every 500 points.*

- **Losing Lives:**

if pygame.sprite.spritecollide(player, asteroids, True):

*Lose 1 life upon asteroid collision.*

lives -= 1

## 3. Up and Down Movement

- **Unlocked at 300 Points:**

if score >= 300 and not player.has\_vertical\_movement:

player.has\_vertical\_movement = True

player.speed\_y = 5

*Enables vertical movement  
(up/down) after reaching 300  
points.*

## 4. Asteroids Scaling Over Time

- **Size Scaling:**

```
size_scale = min(1 + (game_time / 60) * 0.1, 2.0)
```

- Asteroids grow up to **2x size** as time increases.

- **Speed Scaling:**

```
speed_scale = min(1 + (game_time / 60) * 0.05, 1.5)
```

```
self.speed = ASTEROID_SPEED * (50 / scaled_size) * speed_scale
```

- Speed increases with time, making them faster and harder to dodge.

- **Spawn Frequency:**

```
base_spawn_interval = 0.4
```

```
current_interval = max(base_spawn_interval - (game_time / 30) * 0.1, 0.15)
```

- Asteroids spawn **faster** as time progresses.

## 5. Score Calculation

- **Destroy Asteroids:**

```
hits = pygame.sprite.groupcollide(bullets, asteroids, True, True)
```

```
score += 10
```

- Gain **10 points** per asteroid destroyed.

- **Collect Crystals:**

```
crystal_sound.play()
```

```
score += 10
```

- Gain **10 points** per crystal collected.

- **High Score:**

```
high_score = max(high_score, score)
```

- Tracks the **highest score** achieved.

## 6. Main Function (main())

- **Initial Setup:**

```
player = Player()
all_sprites.add(player)
background = Background()
```

- Creates the **player** and **background** objects.



- **Game Loop:**

```
while running:
    # Control game speed
    clock.tick(FPS)

    all_sprites.update()
    background.update()
```

- Continuously **updates all objects** and background animation.

- **Spawning Objects:**

```
if spawn_timer >= current_interval:
    spawn_timer = 0
    asteroid_chance = min(0.9 + (game_time / 30) * 0.02, 0.98)
    spawn_count = min(1 + int(game_time / 60), 3)

    for _ in range(spawn_count):
        if random.random() < asteroid_chance:
            asteroid = Asteroid(game_time)
            asteroids.add(asteroid)
            all_sprites.add(asteroid)
        else:
            crystal = Crystal()
            crystals.add(crystal)
            all_sprites.add(crystal)
```

- Spawns **asteroids and crystals** dynamically based on elapsed time.

- **Collision Handling:**

```
crystal_hits = pygame.sprite.spritecollide(player, crystals, True)
for crystal in crystal_hits:
    crystal_sound.play()
    score += 10
    crystal_count += 1
    high_score = max(high_score, score)

    # Power-up activation every 5 crystals collected
    if crystal_count % 5 == 0:
        player.activate_power_up()
        active_messages.append(show_message(screen, "Power-up activa

    # Unlock vertical movement at score 300
    if score >= 300 and not player.has_vertical_movement:
        player.has_vertical_movement = True
        player.speed_y = 5
        active_messages.append(show_message(screen, "Vertical moveme

# Bullet hits on asteroids
hits = pygame.sprite.groupcollide(bullets, asteroids, True, True)
```

- Checks collisions for **crystals, asteroids, and bullets**.

- **Game Over Handling:**

```
if lives <= 0:
    game_over = True
```

- Ends the game when lives reach **0** and displays a **restart prompt**.

## Conclusion

"Space Scavenger" offers an engaging experience with its dynamic gameplay and scaling difficulty. The combination of responsive controls, power-ups, and immersive sound effects provides a challenging and rewarding space adventure.