MINI PROJECT REPORT

GAME DOVLEPMENT USING MUTUAL EXCLUSION

21CSC202J OPERATING SYSTEMS

Submitted by

Kollu Nagakiran Kumar [RA2211028010230]

Sai Dinesh Papala [RA2211029010016]

Rahul Chowdhary [RA2211029010024]

Under the Guidance of

DR.P. MAHALAKSHMI

Assistant Professor, Department of Computing Technologies

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR-603203

NOVEMBER 2023

BONAFIDE CERTIFICATE

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956) BONAFIDE CERTIFICATE

Certified that the Mini project report titled **"GAME DEVELOPMENT USING MUTUAL EXCLUSION"**is the Bonafide work of **SAI DINESH PAPPALA[RA2211029010016], RAHUL [RA2211029010024], KOLLU NAGAKIRAN[RA2211028010230]** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

DR. P. MAHALAKSHMI

FACULTY

Assistant Professor

Department of Computing Technologies

ACKNOWLEDGEMENT

We express our humble gratitude to Dr. C. Muthamizhchelvan, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, Dr. T.V.Gopal, for his invaluable support.

We wish to thank Dr. Revathi Venkataraman, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We are incredibly grateful to our Head of the Department, Dr. M. Lakshmi, Professor, Department of Data Science and Business Systems, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our Project Coordinator, Dr. P. Rajasekar, Assistant Professor, Panel Head, Dr. A. Murugan, Associate Professor and members, Dr. R. Rajkumar, Assistant Professor, Dr. T. Karthick, Assistant Professor and Dr. S. Jeeva, Assistant Professor, Department of Data Science and Business Systems, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor, Dr. S. Ushasukhanya, Assistant Professor, Department of Networking and Communication, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, Dr. A. Murugan, Associate Professor, Department of Data Science and Business Systems, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under his mentorship. He provided us with the freedom and support to explore the research topics of our interest. His passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank the Data Science and Business Systems Department staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

DEPARTMENT OF
SCHOOL OF COMPUTING
College of Engineering and Technology
SRM Institute of Science and Technology

MINI PROJECT REPORT

ODD Semester, 2023-2024

Lab code & Sub Name : 21CSS201T & Computer Organization and Architecture

Year & Semester : II & III

Project Title : GAME DEVELOPMENT USING MUTUAL EXCLUSION

Lab Supervisor **: Dr.R.Vidhya**

Team Members : 1. Kollu Nagakiran Kumar (RA2211028010230)
2.Sai Dinesh Papala (RA2211029010016)
3.Rahul Chowdhary(RA2211029010024)

Particulars	**Max. Marks**	**Marks Obtained**
		Name:
		Register No :
Program and Execution	20	
Demo verification &viva	15	
Project Report	05	
Total	**40**	

Date : 14-11-2023
Staff Name : Dr.Mahalakshmi Pushparaja
Signature :

TABLE OF CONTENT:

GAME DEVELOPMENT USING MUTUAL EXCLUSION

OBJECTIVE:

GAME DEVELOPMENT USING MUTUAL EXCLUSION

OBJECTIVE:

The objective for game development using mutual exclusion primarily revolves around ensuring that shared resources or critical sections of code are accessed in a way that prevents conflicts and maintains data integrity. Here are some specific objectives:

1. Concurrency Control:
   - Implement mutual exclusion mechanisms to control access to shared resources in a multi-threaded or multi-process game environment.
   - Ensure that only one thread or process can access critical sections of code at a time to prevent data corruption or unpredictable behavior.

2. Thread Safety:
   - Design and implement game systems with thread safety in mind to avoid race conditions and deadlocks.
   - Utilize mutual exclusion techniques such as locks, semaphores, or mutexes to synchronize access to shared data structures.

3. Resource Management:
   - Safeguard shared resources such as game state, player data, or graphics buffers by employing mutual exclusion.
   - Implement efficient resource locking strategies to minimize contention and maximize performance.

4. Data Consistency:
   - Guarantee the consistency of game data by enforcing mutual exclusion in areas where multiple threads or processes can potentially modify the same data concurrently.
   - Use atomic operations or transactional memory in critical sections to maintain data consistency.

5. Deadlock Prevention:
   - Design systems with deadlock prevention mechanisms to avoid situations where multiple threads are blocked indefinitely, waiting for each other.
   - Implement timeout strategies or use resource hierarchy to prevent and resolve deadlocks.

6. Performance Optimization:
   - Fine-tune mutual exclusion mechanisms to strike a balance between thread safety and performance.
   - Consider the granularity of locks to minimize contention and increase parallelism where possible

ABSTRACT:

This paper explores the design and development aspects of a classic yet enduring genre in casual gaming – the endless runner – through the lens of a popular exemplar, the Dino Game. Originating from the 'offline' screen in Google Chrome, the Dino Game has evolved into a cultural icon, captivating players with its simple mechanics and addictive gameplay.

The paper delves into the game's foundational principles, dissecting the essential elements that contribute to its universal appeal. It analyzes the minimalist yet effective visual design, highlighting the significance of simplicity in fostering accessibility and engagement. The straightforward controls, limited to jumping and ducking, are discussed in terms of creating an intuitive user experience that transcends language and age barriers.

Moreover, the paper examines the procedural generation of obstacles and the game's adaptive difficulty curve, crucial components that ensure sustained player interest and challenge. It explores how randomness in obstacle placement contributes to the game's replayability, keeping the experience fresh and unpredictable for players across countless sessions.

The Dino Game's incorporation of a scoring system is also scrutinized, emphasizing how this seemingly basic feature adds depth to the gameplay, transforming a simple survival mechanic into a competitive pursuit. The role of achievements and leaderboards in enhancing player motivation and fostering a sense of community is discussed in the context of modern gaming trends.

In addition to dissecting the game's mechanics, the paper delves into the broader implications of the Dino Game as a cultural phenomenon and its impact on the gamification of internet downtime. It explores how such games have become a means of turning moments of frustration or connectivity issues into opportunities for entertainment and stress relief.

The conclusion reflects on the enduring success of the Dino Game and its influence on game design paradigms. It offers insights for developers seeking to create engaging and accessible games, emphasizing the importance of balancing simplicity, challenge, and community aspects. The Dino Game stands as a testament to the timeless allure of well-crafted gameplay, proving that even the most straightforward concepts can leave a lasting mark on the gaming landscape.

INTRODUCTION:

In the vast realm of programming, game development stands as a captivating intersection of creativity and technical prowess. PyCharm, a powerful integrated development environment (IDE) for Python, provides an ideal platform for crafting engaging and interactive games. In this tutorial, we embark on a journey to create a simple yet exciting game – the Dino Game – using PyCharm.

The Dino Game, popularized by its inclusion in the Google Chrome browser as a hidden Easter egg, features a dinosaur navigating through obstacles in a prehistoric landscape. Our goal is to introduce you to the basics of game development using PyCharm, guiding you through the process of setting up your project, creating game elements, implementing interactivity, and handling game mechanics.

Setting the Stage

We'll start by setting up a Python project in PyCharm and configuring the necessary dependencies. Pygame, a cross-platform set of Python modules designed for game development, will be our toolkit of choice for this adventure.

Creating the Dino Character

Every game needs a protagonist, and in our case, it's a pixelated dinosaur. We'll delve into the basics of sprite creation, animation, and character movement using Pygame.

Designing the Prehistoric Landscape

To bring our game to life, we'll design a simple prehistoric landscape with obstacles that our dino must navigate. This involves creating backgrounds, platforms, and dynamically moving obstacles.

Implementing Interactivity

Games thrive on user interaction. We'll explore how to capture keyboard inputs to make the dino jump and avoid obstacles. Pygame's event handling mechanisms will be crucial here.

Adding Challenges and Scoring

No game is complete without challenges and a way to measure success. We'll incorporate obstacles of varying difficulty levels and implement a scoring system to keep track of the player's accomplishments.

HARDWARE/SOFTWARE REQUIREMENTS:

Game development involves both hardware and software requirements. The specific requirements can vary depending on the complexity and platform of the game you are developing. Here are some general guidelines for both hardware and software:

Hardware Requirements:

1. Processor (CPU):
   - Multi-core processors are recommended for handling the complex calculations involved in game development.
   - Intel Core i5 10^{th} gen

2. Memory (RAM):
   - 16 GB , as game development tools and engines can be memory-intensive.

3. Graphics Card (GPU):
   - A dedicated graphics card is essential, especially for 3D game development.
   - VR development may require a more powerful GPU.

4. Storage:
   - 1 TB SSDs (Solid State Drives) are preferred for faster data access and shorter load times.
   - Adequate storage space for the game assets and development tools.

5. Monitor;
   - A high-resolution monitor for better visibility and workspace. 1920x1080

6. Input Devices:
   - A comfortable keyboard and mouse or other input devices.
   - Graphic tablets for artists working on game art.

7. Internet Connection:
   - A stable internet connection is essential for accessing updates, online resources, and collaborative work.

Software Requirements:

1. Integrated Development Environment (IDE):
   - We used Pycharm Studio, Ursina Engine, Game Development environments.

2. Version Control System;

   - Adobe Creative Cloud (Photoshop, Illustrator, etc.) for 2D art.

3. Sound Design:
Software like Audacity, Adobe Audition, or other digital audio workstations (DAWs) for sound editing.

4. Programming Languages:
   - C# is commonly used with Unity, while C++ is prevalent with Unreal Engine.
   - Python and other scripting languages may be used for various tasks.

CONCEPTS/WORKING PRINCIPLE

Seps for implementing mutual exclusion in a Dino Game:

1. Identify Shared Resources:

- Identify the variables or data that multiple parts of your game might access or modify concurrently, such as scores or game state.

2. Choose a Synchronization Mechanism:

- Pick a synchronization tool like locks or semaphores to control access to shared resources.

3. Implement Locking Mechanisms:

- Use locks to protect critical sections of your code. Acquire the lock before entering a critical section and release it afterward.

4. Apply Locks to Game Logic:

- Identify parts of your Dino Game code where shared resources are accessed and apply locks to those sections.

5. Test for Concurrency Issues:

- Test your game with multiple threads or players to identify and fix any issues related to concurrent access.

6. Optimize and Refine:

- Continuously refine your implementation for efficiency. Consider using finer-grained locks and optimize critical sections.

Creating a game using Python in PyCharm involves several steps. Below is a basic outline to get you started. I'll use the Pygame library for simplicity, but there are other game development libraries you can explore as well.

Step 1: Install Pygame

Before you start, you need to install the Pygame library. Open a terminal or command prompt and run:

Step 2: Set Up PyCharm

If you haven't already, download and install PyCharm from the official website. Open PyCharm and create a new Python project.

Step 3: Create a Python File

In PyCharm, right-click on your project folder, select **New**, and then choose **Python File**. Give your file a name (e.g., **game.py**).

Step 4: Write Your Game Code

Write the code for your game. Here's a simple example using Pygame:

Step 5: Run Your Game

Click the green "Run" button in PyCharm to execute your game. You should see a window with a blue rectangle.

Step 6: Expand Your Game

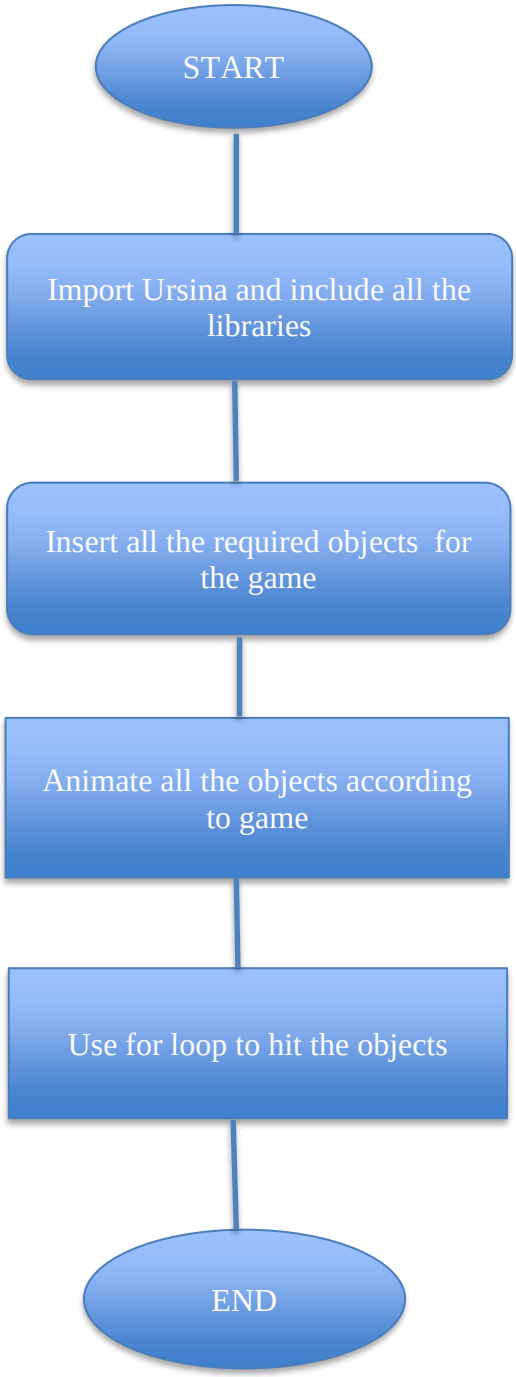
Continue building your game by adding more features, images, sounds, and logic. Pygame has extensive documentation that can guide you: Pygame Documentation.

APPROACH/METHODOLOGY/PROGRAMS:

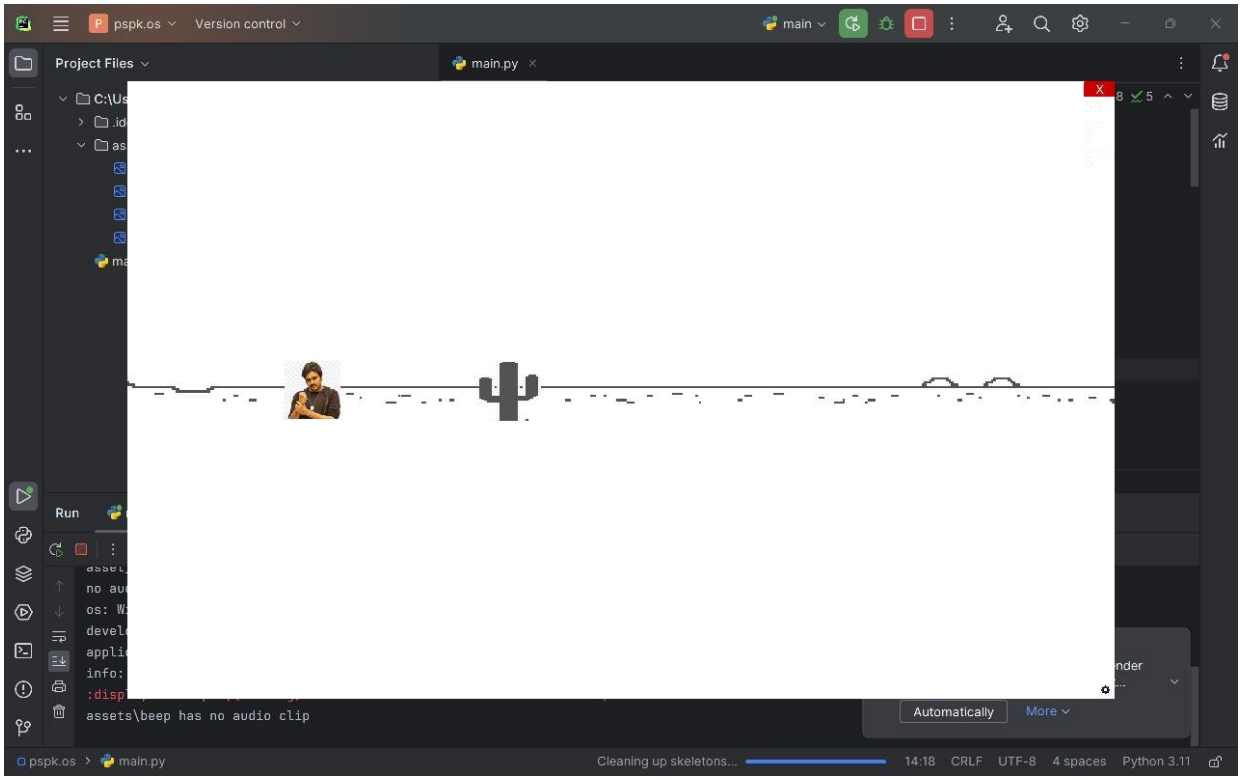
```
import timeit

from ursina import *
import random as r

app = Ursina()
window.color = color.white


dino = Animation('assetss\dino',
                 collider='box',
                 x=-5)

ground1 = Entity(
  model='quad',
  texture='assetss\ground',
  scale=(50,0.5,1),
  z=1)

ground2 = duplicate(ground1, x=50)
pair = [ground1, ground2]

cactus = Entity(
  model='quad',
  texture='assetss\cacti',
  x = 20,
  collider='box'
)
cacti = []
def newCactus():
  new = duplicate(cactus,
                  x=12+r.randint(0,5))
  cacti.append(new)
  invoke(newCactus, delay=2)

newCactus()

def update():
    for ground in pair:
        ground.x -= 6*time.dt
        if ground.x < -35:
            ground.x += 100
    for c in cacti:
        c.x -= 6 * time.dt
    if dino.intersects().hit:
```

```
        dino.texture = 'assets\hit'
        application.pause()

sound = Audio(
  'assets\\beep',
  autoplay=False
)


def input(key):
  if key == 'space':
    if dino.y < 0.01:
      sound.play()
      dino.animate_y(
        2,
        duration=0.4,
        curve= curve.out_sine
      )
      dino.animate_y(
        0,
        duration=0.4,
        delay=0.4,
        curve = curve.in_sine
      )



camera.orthographic = True
camera.fov = 10

app.run()
```

FLOWCHART:


START
Import Ursina and include all the libraries
Insert all the required objects for the game
Animate all the objects according to game
Use for loop to hit the objects
END

OUTPUT:


pspk.os
Version control
main
Project Files
main.py
Run
assets\beep has no audio clip
Automatically
More
pspk.os
main.py
Cleaning up skeletons...
14:18
CRLF
UTF-8
4 spaces
Python 3.11

LITERATURE SURVEY:

- Book Title: “Operating System Concepts"
- Authors: ABRAHAM SILBERSCHATZ, PETER BAER GALVIN, GREG GAGNE Edition: TENTH
- Publisher: WILEY
- We referred to this book to understand what is Mutual Exclusion, Critical Section and how to use these concepts to write a code for a game
  4. Browser Version:
- - The Dino Game may have specific optimizations for certain versions of the Chrome browser. It's recommended to use an up-to-date version of Google Chrome for the best experience.
  5. Hardware:
- - The Dino Game is lightweight and does not require high-end hardware. It should run smoothly on most modern computers, laptops, and even on some mobile devices.
  6. Browser Settings:
- - Ensure that JavaScript is enabled in your browser settings, as the game relies on this scripting language for its functionality.
  7. Graphics Card:
- The Dino Game does not have high graphics demands, and it can run on systems with integrated graphics or basic dedicated graphics cards.
- Any modern graphics card capable of rendering 2D graphics should be sufficient.

CONCLUSIONS:

In the realm of game development, the PyCharm Dino Game stands as a testament to the fusion of creativity and code. Leveraging the capabilities of PyCharm, this project encapsulates the essence of simplicity and engagement, providing developers with a canvas to explore both the artistry of design and the precision of programming.

The journey began with the inception of the idea – a classic endless runner featuring a pixelated dinosaur navigating a dynamically generated landscape. PyCharm, with its powerful integrated development environment (IDE), offered an intuitive platform for coding, debugging, and testing. The Python programming language served as the backbone, enabling developers to craft an interactive and responsive gaming experience.

As the development process unfolded, the significance of PyCharm's features became apparent. The IDE's intelligent code completion, debugging tools, and version control integration streamlined the coding workflow, empowering developers to focus on refining gameplay mechanics and enhancing user experience. The synergy between PyCharm and Python facilitated rapid prototyping, allowing for agile iterations and creative experimentation.

The incorporation of game physics, collision detection, and animation within PyCharm showcased the versatility of the IDE in handling multimedia elements. The Dino Game's minimalist aesthetic and smooth gameplay underscored the balance between visual appeal and technical functionality achievable through PyCharm.

Collaboration and community engagement were integral to the project's success. PyCharm's support for collaborative development and its vibrant community contributed to the exchange of ideas, code snippets, and solutions. The Dino Game, evolving through shared knowledge and collaborative effort, exemplifies the cooperative spirit fostered by PyCharm.

In conclusion, the PyCharm Dino Game project exemplifies the synergy between creativity and code, facilitated by a robust development environment. PyCharm's feature-rich IDE, coupled with the flexibility of Python, allowed developers to bring a captivating and enjoyable game to life.

REFERENCES:

Game Development Books –

Titles:"GameEngine Architecture" by Jason Gregory can be helpful.

Paper Title: “A Review of various Mutual Exclusion Algorithms in Distributed Environment”
Authors: NISHA YADAV, SUDHA YADAV, SONAM MANDIRATTA