

PROFESSIONAL TRAINING REPORT

at

**Sathyabama Institute of Science and Technology
(Deemed to be University)**

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Technology Degree in

Information Technology

By

**R.VEDHANAYAKI
(Reg.No.37120082)**



**DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING**

**SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119**

AUGUST 2020



SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

(Established under Section 3 of UGC Act, 1956)
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai -600119
www.sathyabama.ac.in



DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **R.VEDHANAYAKI** (Reg.no.37120082) who carried out the project entitled “**CAR GAME USING PYTHON PYGAME**” under our supervision from DEC 2020 to JAN 2021.

Internal Guide

E.BRUMANCIA M.E,(Ph.D.,)

Head of the Department

Dr.R SUBHASHINI M.E,Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I , R.VEDHANAYAKI (Reg.No.37120082) hereby declare that the Project Report entitled “**CAR GAME USING PYTHON PYGAME**” done by me under the guidance of **E.BRUMANCIA M.E(Ph.D)**., is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in Information Technology .

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D, Dean**, School of Computing , **Dr.SUBHASHINI M.E., Ph.D.** , Head of the Department of Information and Technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **E.Brumancia M.E,(Ph.D.,)** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.



HTC Global Services (India) Private Ltd.

SDF II, Phase II, MEPZ, Tambaram, Chennai-600 045, INDIA
Tel : +91 44 2262 3522 Fax : +91 44 2262 7713
Web : www.htcinc.com | CIN: U72900TN2001PTC047862

July 05, 2019

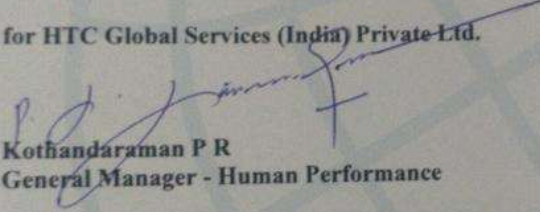
TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Ms. Vedhanayaki R** has successfully completed her summer internship program with us from 11th November, 2018 to 12th December, 2018.

During the internship tenure, she worked on the domain **R data Science and Networking Tools** under the guidance of Dr. Muthukumaran Balasubramanian A J, General Manager - Training, Institute of Technology Management & Research, a group of HTC Global Services. During the tenure, her conduct, attitude and performance were found to be good.

We wish her all the very best in her future endeavors.

for HTC Global Services (India) Private Ltd.


Kothandaraman P R
General Manager - Human Performance

World Headquarters

3270 West Big Beaver Road, Troy, MI 48064, USA
Phone: 248.786.2500 • Fax: 248.786.2515 • www.htcinc.com
India • Malaysia • Singapore • UAE • Australia • Indonesia • S



07/12/2020

Vedhanayaki R

has successfully completed

Blockchain Basics

an online non-credit course authorized by University at Buffalo and The State University of New York and offered through Coursera

Bina Ramamurthy
Teaching Professor
Computer Science and Engineering

**COURSE
CERTIFICATE**



Verify at coursera.org/verify/QCB9SYXVTXD7
Coursera has confirmed the identity of this individual and
their participation in the course.

ABSTRACT

The project titled Game Development Using Python Pygame. Pygame is a cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language. Pygame uses the Simple Direct Media Layer (SDL) library, with the intention of allowing real-time computer game development without the low-level mechanics of the C programming language and its derivatives. This is based on the assumption that the most expensive functions inside games can be abstracted from the game logic, making it possible to use a high-level programming language, such as Python, to structure the game. Other features that SDL doesn't have include vector math, collision detection, 2d sprite scene graph management, MIDI support, camera, pixel-array manipulation, transformations, filtering, advanced freetype font support, and drawing. Applications using pygame can run on Android phones and tablets with the use of pygame Subset for Android (pgs4a) Sound, vibration, keyboard, and accelerometer are supported on Android. Pygame is suitable to create client-side applications that can be potential.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE.NO
1	ABSTRACT LIST OF ABBREVIATIONS LIST OF FIGURES INTRODUCTION 1.1 INTRODUCTION TO THE PROJECT 1.2 AIM AND SCOPE 1.3 FEASIBILITY STUDY 1.4 SYSTEMS	 1 1 1 3
2	ALGORITHMS AND METHODS 2.1 SDLC METHODOLOGY 2.2 ACTIVITY DIAGRAM	 4 6
3	SYSTEM DESIGN AND TESTING PLAN 3.1 MODULES 3.2 INPUT DESIGN 3.3 DATA FLOW DIAGRAM 3.4 UML DIAGRAM 3.5 TEST PLAN 3.6 SYSTEM REQUIREMENTS 3.7 NON-FUNCTIONAL REQUIREMENTS	 7 8 10 11 15 19 19
4	RESULTS AND DISCUSSIONS 4.1 SUMMARY 4.2 SCREEN SHOTS	 22 23
5	CONCLUSION AND FUTURE ENHANCEMENTS 5.1 CONCLUSION 5.2 FUTURE ENHANCEMENT 5.3 REFERENCES	 36 37 37
6	APPENDIX 6.1 SOURCE CODE	 38

LIST OF FIGURES

FIGURE NO.	TITLE OF THE FIGURE	PAGE NO.
2.1	Activity Diagram	6
3.1.2	System Architecture	7
3.3	Data Flow Diagram	9
3.4.1	Use Case Diagram	10
3.4.2	Class Diagram	11
3.4.3	Sequence Diagram	12
3.4.4	Component Diagram	13

LIST OF ABBREVIATIONS

ACRONYM

DFD

UML

EPL

JDT

IDE

GPS

EXPANSION

Data Flow Diagram

Unified Modeling Language

Eclipse Public License

Java Development Tools

Integrated Development
Environment

Global Positioning System

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO THE PROJECT

pygame is a Python wrapper for the SDL library, which stands for Simple DirectMedia Layer. SDL provides cross-platform access to your system's underlying multimedia hardware components, such as sound, video, mouse, keyboard, and joystick. pygame started life as a replacement for the stalled PySDL project. The cross-platform nature of both SDL and pygame means you can write games and rich multimedia Python programs for every platform that supports them.

1.2. AIM AND SCOPE:

1.2.1. AIM:

The main aim of the project is "To Create Car Game Using Pygame Python".

1.2.2. SCOPE:

The pygame library is composed of a number of Python constructs, which include several different modules. These modules provide abstract access to specific hardware on your system, as well as uniform methods to work with that hardware. For example, display allows uniform access to your video display, while joystick allows abstract control of your joystick.

1.2.3. Feasibility Study

1.2.4. Technical Feasibility

pygame also includes several Python classes, which encapsulate non-hardware dependent concepts. One of these is the Surface which, at its most basic, defines a rectangular area on which you can draw. Surface objects are used in many contexts in pygame. Later you'll see how to load an image into a Surface and display it on the screen. In pygame, everything is viewed on a single user-created display, which can be a window or a full screen. The display is created using `.set_mode()`, which returns a Surface representing the visible part of the window. It is this Surface that you pass into drawing functions like `pygame.draw.circle()`, and the contents of that Surface are pushed

organization currently possesses the required technologies. Is the required technology available with the organization?

1.2.5. Operational Feasibility

Before you start writing any code, it's always a good idea to have some design in place. Since this is a tutorial game, let's design some basic gameplay for it as well:

The goal of the game is to avoid incoming obstacles:

- The player starts on the left side of the screen.
- The obstacles enter randomly from the right and move left in a straight line.
- The player can move left, right, up, or down to avoid the obstacles.
- The player cannot move off the screen.
- The game ends either when the player is hit by an obstacle or when the user closes the window.
- No multiple lives
- No scorekeeping
- No player attack capabilities
- No advancing levels
- No boss characters

1.3.3.Economic Feasibility

Economic feasibility attempts to weigh the costs of developing and implementing a new system, against the benefits that would accrue from having the new system in place. This feasibility study gives the top management the economic justification for the new system.

A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. In addition, this proves to be a useful point of reference to compare actual costs as the project progresses. There could be various

types of intangible benefits on account of automation. These could include increased customer satisfaction, improvement in product quality better decision making timeliness of information, expediting activities, improved accuracy of operations, better documentation and record keeping, faster retrieval of information, better employee morale.

1.4 SYSTEMS:

1.4.1 EXISTING SYSTEM

The existing system consists of entering the details in the Microsoft Excel Sheets for the storing of the data. When a manager needs information of the employee he searches for the specified file in the file system. He opens the file and takes the information. Report Generation done manually by copying the content of the different files into another file. The Manually generated report was then printed.

DISADVANTAGES OF EXISTING SYSTEM :

- Information retrieval is a very big process.
- Lack of organization of the files may lead to information loss due to accidental deletion of files.
- No security because the files are visible to the users.
- Report generation will be a big task.

1.4.2 PROPOSED SYSTEM

The Proposed system is a browser which is completely related to online system, which provides the centralized database. It stores Defects data and description of the particular Defect data. It can also create reports and documents based on the information in its database.

ADVANTAGES OF PROPOSED SYSTEM :

- The performance is increased due to well designed database.
- Security is increased
- Time saving in report generation
- Easy to update the details

CHAPTER 2

ALGORITHM AND METHODS

2.1. SDLC METHODOLOGY:

This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process. SPIRAL MODEL was defined by Barry Boehm in his 1988 article, "A spiral Model of Software Development and Enhancement.

As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

The steps for Spiral Model can be generalized as follows:

- The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
- A preliminary design is created for the new system.
- A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
- A second prototype is evolved by a fourfold procedure:
 1. Evaluating the first prototype in terms of its strengths, weakness, and risks.
 2. Defining the requirements of the second prototype.
 3. Planning an designing the second prototype.
 4. Constructing and testing the second prototype.

- At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involved development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer's judgment, result in a less-than-satisfactory final product.
- The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
- The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
- The final system is constructed, based on the refined prototype.
- The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

The following diagram shows how a spiral model acts like:

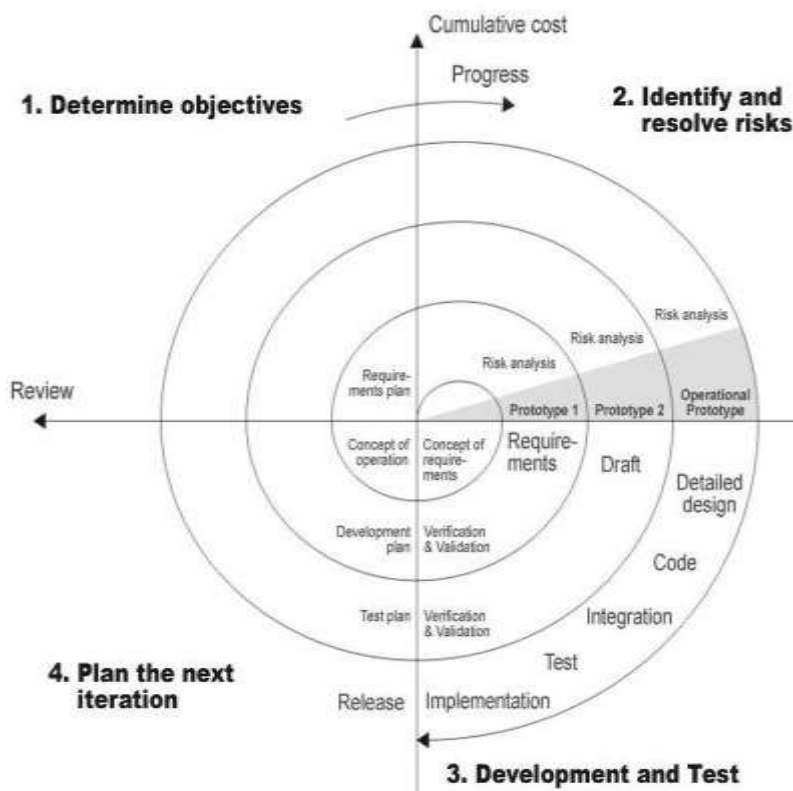


Fig 1.0-Spiral Model

ADVANTAGES

- Estimates(i.e. budget, schedule etc .) become more realistic as work progresses, because important issues discovered earlier.
- It is more able to cope with the changes that are software development generally entails.
- Software engineers can get their hands in and start working on the core of a project earlier.

2.2 ACTIVITY DIAGRAM:

ACTIVITY DIAGRAM

Admin:

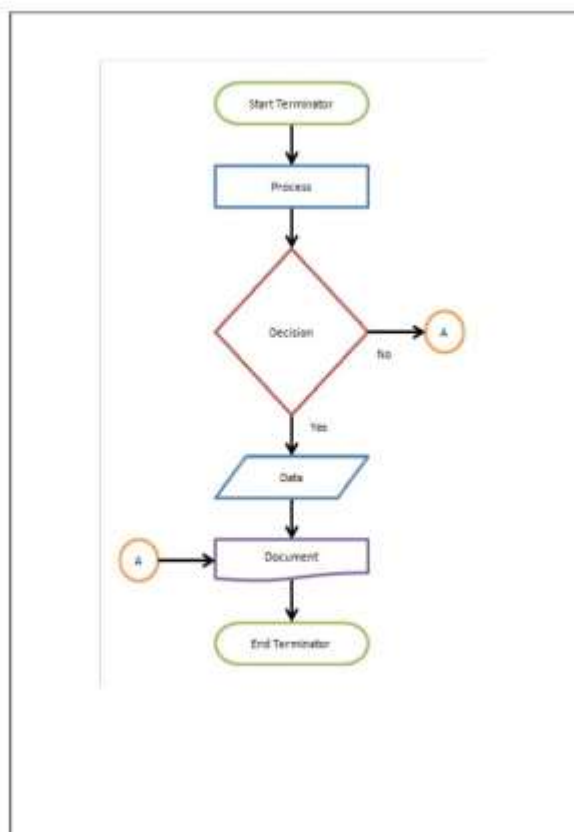


Fig 2.1. Activity diagram

CHAPTER 3

SYSTEM DESIGN AND TESTING PLAN

3.1.1 MODULES:

ADMIN: This module has the entire access to all other modules, admin creates the project and assigning the projects to the created manager, adding members to the managers, assigning Defects based on the priority. Can update the manager, members and access to the particular project data. Generating reports based on the managers report submission.

Manager: Manager has the full access to the particular project assigned by the admin and controls the team members access to the Defects assigned. Has the permission to generate the reports and update the information of team members and adding members to the project.

Developer: Can access the task or Defect assigned by the manager, view assigned projects and resolving the assigned Defect. Developer can view the Defects list assigned by the manager.

Tester: Tester can access to the projects , can view the assigned projects and can add a new Defect to the list and send the bug back to the manager. Tester can login to the system and access the assigned projects list.

Reports: Both Admin and Manager can access this module and generate the reports based on the requirements.

3.1.2 SYSTEM ARCHITECTURE

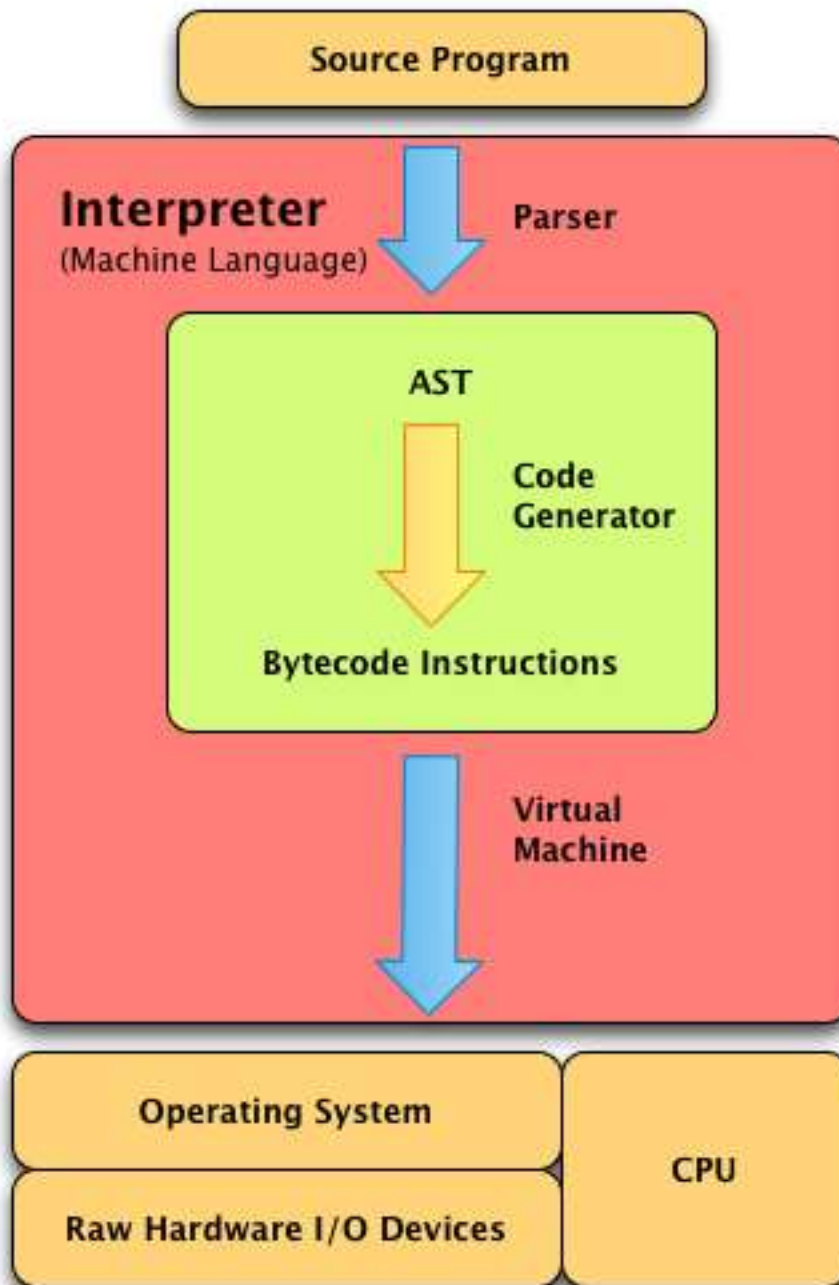


Fig .3.1.2 shows system architecture

3.2 INPUT DESIGN:

3.2.1. THE PRESENTATION LAYER

It Also called as the client layer comprises of components that are dedicated to presenting the data to the user. For example: Windows/Web Forms and buttons, edit boxes, Text boxes, labels, grids, etc.

3.2.2. THE BUSINESS RULES LAYER

This layer encapsulates the Business rules or the business logic of the encapsulations. To have a separate layer for business logic is of a great advantage. This is because any changes in Business Rules can be easily handled in this layer. As long as the interface between the layers remains the same, any changes to the functionality/processing logic in this layer can be made without impacting the others. A lot of client-server apps failed to implement successfully as changing the business logic was a painful process

3.2.3. THE DATA ACCESS LAYER

This layer comprises of components that help in accessing the Database. If used in the right way, this layer provides a level of abstraction for the database structures. Simply put changes made to the database, tables, etc do not affect the rest of the application because of the Data Access layer. The different application layers send the data requests to this layer and receive the response from this layer.

3.2.4. THE DATABASE LAYER

This layer comprises of the Database Components such as DB Files, Tables, Views, etc. The Actual database could be created using SQL Server, Oracle, Flatfiles,etc.

In an n-tier application, the entire application can be implemented in such a way that it is independent of the actual Database. For instance, you could change the Database Location with minimal changes to Data Access Layer. The rest of the Application should remain unaffected.

3.3. DATA FLOW DIAGRAM (DFD)

A data flow diagram is a way of representing a flow of data of a process or a system. The DFD also provides the information about the input and output of each entity and the process itself.

DFD- LEVEL

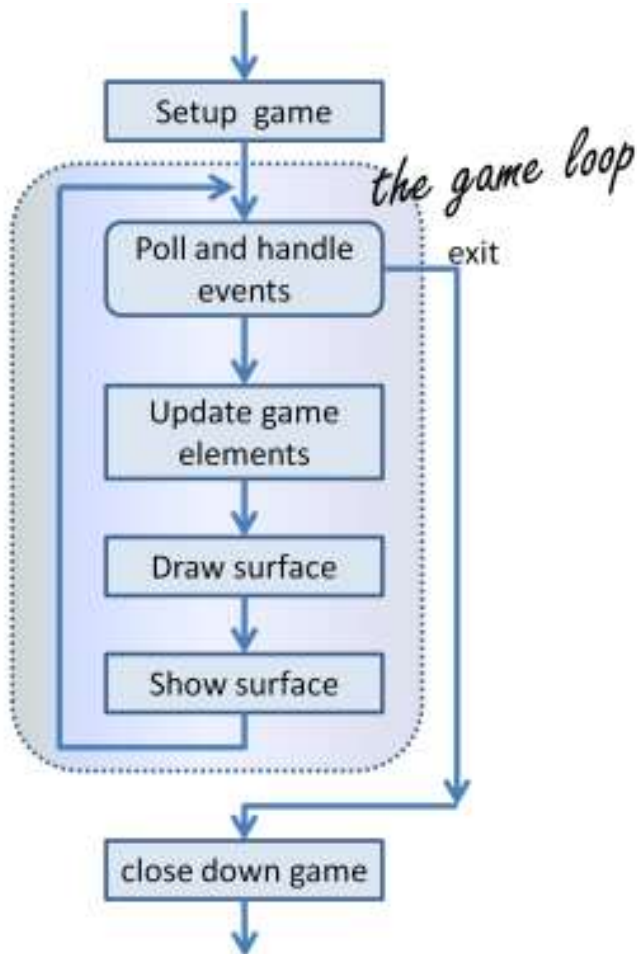


Fig 3.3. shows data flow diagram

3.4. UML DIAGRAMS

Unified Modelling language includes a set of graphic notations techniques to create a visual models of object oriented software intensive systems

3.4.1 TYPES OF UML DIAGRAMS

- Use Case Diagram
- Class diagram
- Sequential diagram
- Component diagram

Use case diagram :

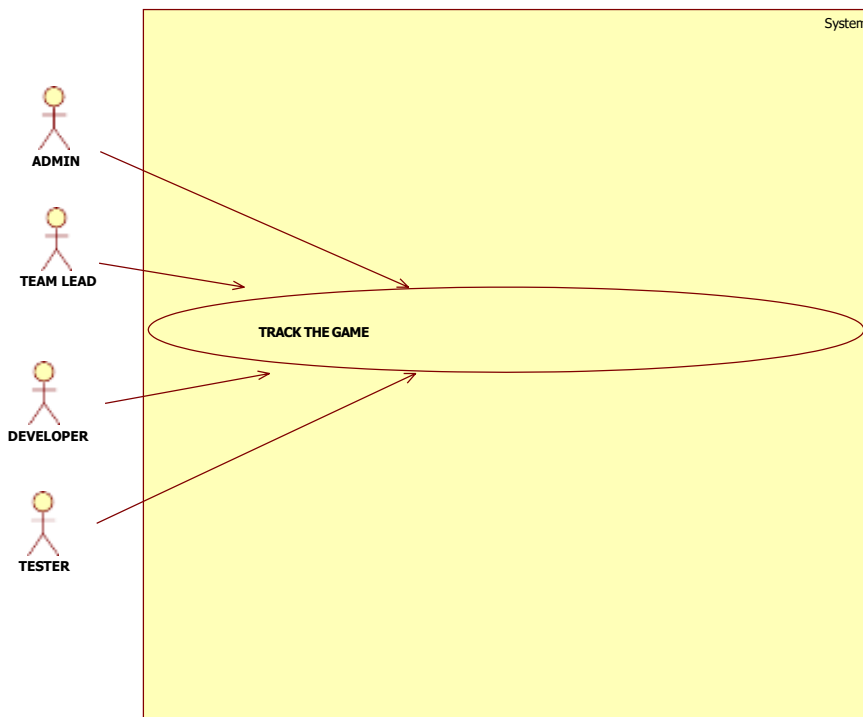


Fig. 3.4.1 Shows use case diagram

CLASS DIAGRAM :

In software Engineering , a class diagram in the unified modeling language is a type of structure diagram that describes the structure of a system by showing the systems ,classes ,their structures,operations and their relationships among objects.

In the diagram , classes are represented with bases which contain three parts.

- The top part contains the name of the class.
- The middle point contains the attributes of the class.
- The bottom part gives the methods or operations in the class can take or undertake .

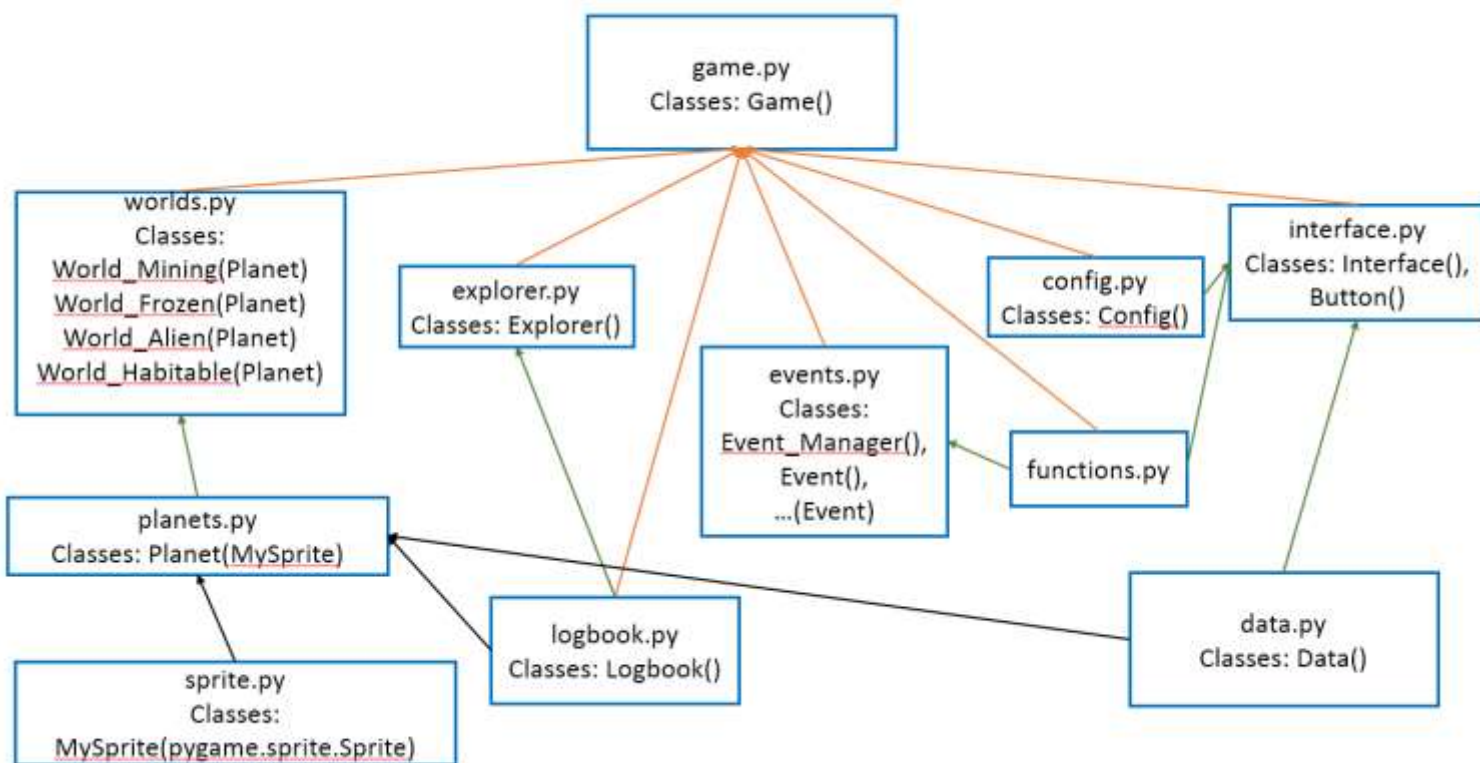


Fig.3.4.1 Shows a class diagram

SEQUENCE DIAGRAM :

- The Sequence diagram shows the ordering of the messages .
- It shows the order of the messages that the admin should perform.
- The interaction that takes place in a collaboration that either realizes a use case or an operation.
- High-level interactions between user of the system and the system,between the system and other systems,or between subsystems.

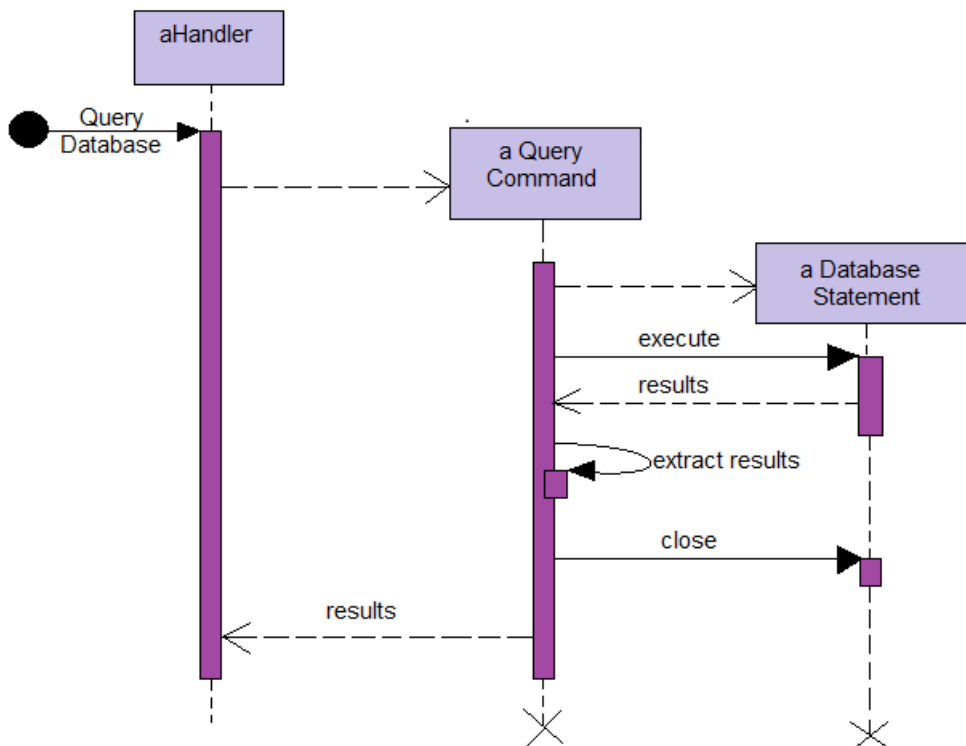


Fig. 3.4.1 Shows sequential diagram

COMPONENT DIAGRAM :

- A component diagram also known as a UML component diagram .
- It describes the organization and wiring of the physical components in a system.
- Component diagrams are often drawn to help model implementation details and double check that every aspect of the systems required.

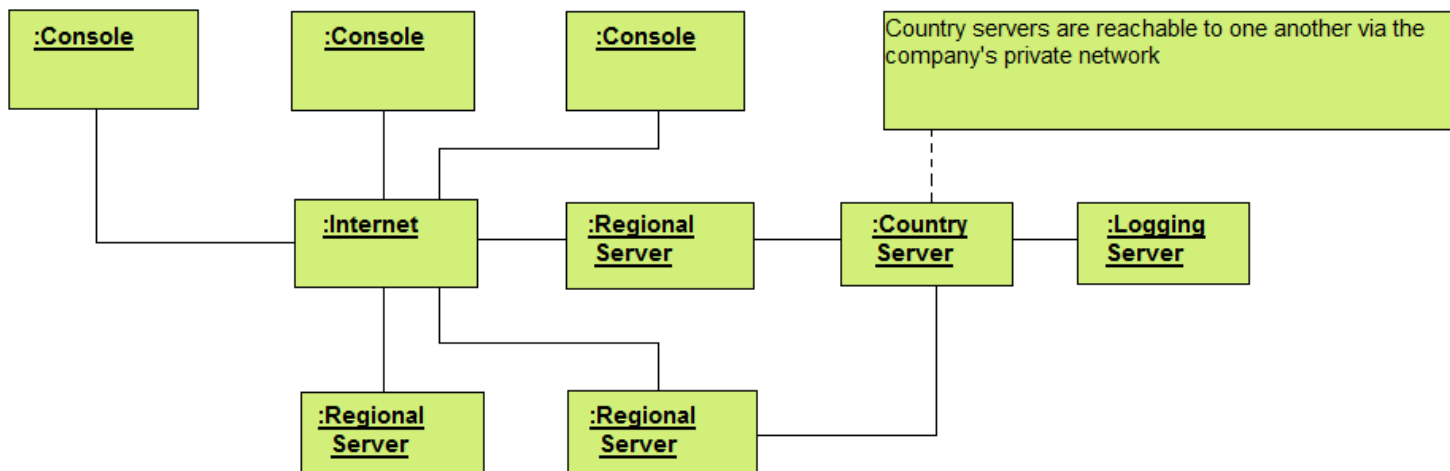


Fig .3.4.1 Shows a component diagram

3.5. TEST PLAN:

- The process of executing a system with the intent of finding an error.
- Testing is defined as the process in which defects are identified, isolated, subjected for rectification and ensured that product is defect free in order to produce the quality product and hence customer satisfaction.
- Quality is defined as justification of the requirements
- Defect is nothing but deviation from the requirements
- Defect is nothing but bug.
- Testing --- The presence of bugs
- Testing can demonstrate the presence of bugs, but not their absence
- Debugging and Testing are not the same thing!
- Testing is a systematic attempt to break a program or the AUT
- Debugging is the art or method of uncovering why the script /program did execute properly.

Testing Methodologies:

- **Black box Testing:** is the testing process in which tester can perform testing on an application without having any internal structural knowledge of application.

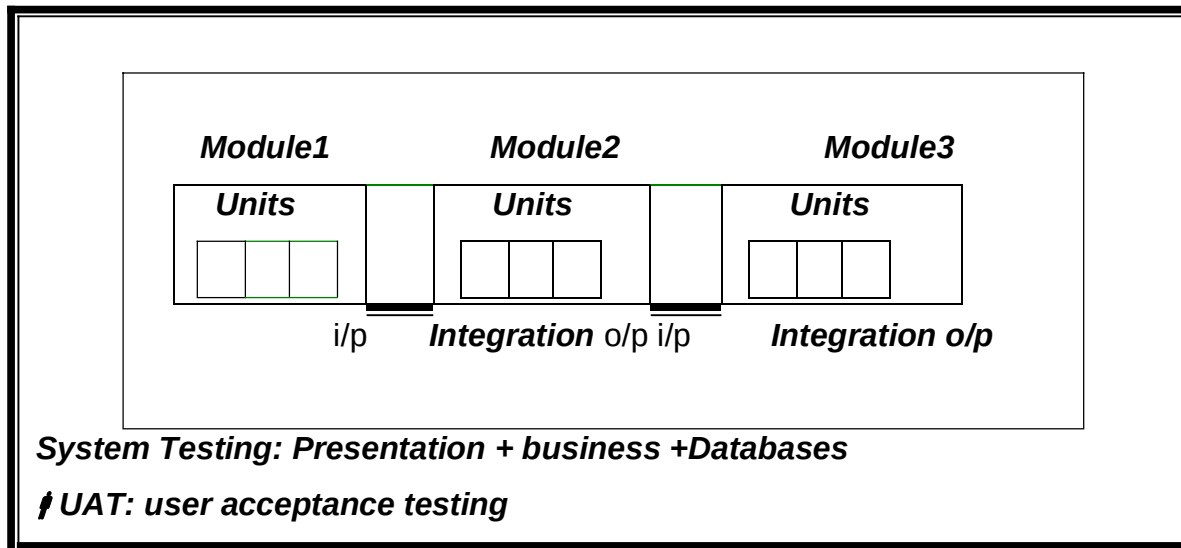
Usually Test Engineers are involved in the black box testing.

- **White box Testing:** is the testing process in which tester can perform testing on an application with having internal structural knowledge.

Usually The Developers are involved in white box testing.

- **Gray Box Testing:** is the process in which the combination of black box and white box tonics' are used.

Levels of Testing:



STLC (SOFTWARE TESTING LIFE CYCLE)

Test Planning:

1. Test Plan is defined as a strategic document which describes the procedure how to perform various testing on the total application in the most efficient way.
2. This document involves the scope of testing,
3. Objective of testing,
4. Areas that need to be tested,
5. Areas that should not be tested,
6. Scheduling Resource Planning,
7. Areas to be automated, various testing tools

Used....

Test Development:

1. Test case Development (check list)
2. Test Procedure preparation. (Description of the Test cases).
3. Implementation of test cases. Observing the result.

Result Analysis:

1. Expected value: is nothing but expected behavior Of application.

2. Actual value: is nothing but actual behavior of application

Bug Tracing: Collect all the failed cases, prepare documents.

Reporting: Prepare document (status of the application)

Types Of Testing:

🌱 > **Smoke Testing:** is the process of initial testing in which tester looks for the availability of all the functionality of the application in order to perform detailed testing on them. (Main check is for available forms)

🌱 > **Sanity Testing:** is a type of testing that is conducted on an application initially to check for the proper behavior of an application that is to check all the functionality are available before the detailed testing is conducted by on them.

🌱 > **Regression Testing:** is one of the best and important testing. Regression testing is the process in which the functionality, which is already tested before, is once again tested whenever some new change is added in order to check whether the existing functionality remains same.

🌱 > **Re-Testing:** is the process in which testing is performed on some functionality which is already tested before to make sure that the defects are reproducible and to rule out the environments issues if at all any defects are there.

🌱 **Static Testing:** is the testing, which is performed on an application when it is not been executed.ex: GUI, Document Testing

🌱 **Dynamic Testing:** is the testing which is performed on an application when it is being executed.ex: Functional testing.

🌱 **Alpha Testing:** it is a type of user acceptance testing, which is conducted on an application when it is just before released to the customer.

🌱 **Beta-Testing:** it is a type of UAT that is conducted on an application when it is released to the customer, when deployed in to the real time environment and being accessed by the real time users.

🌱 **Monkey Testing:** is the process in which abnormal operations, beyond capacity operations are done on the application to check the stability of it in spite of the users abnormal behavior.

🌱 **Compatibility testing:** it is the testing process in which usually the products are

tested on the environments with different combinations of databases (application servers, browsers...etc) In order to check how far the product is compatible with all these environments platform combination.

👤 **Installation Testing:** it is the process of testing in which the tester try to install or try to deploy the module into the corresponding environment by following the guidelines produced in the deployment document and check whether the installation is successful or not.

👤 **Adhoc Testing:** Adhoc Testing is the process of testing in which unlike the formal testing where in test case document is used, with out that test case document testing can be done of an application, to cover that testing of the future which are not covered in that test case document. Also it is intended to perform GUI testing which may involve the cosmetic issues.

TCD (Test Case Document:

Test Case Document Contains

- **Test Scope (or) Test objective**
- **Test Scenario**
- **Test Procedure**
- **Test case**

This is the sample test case document for the Academic details of student project:

Test scope:

- Test coverage is provided for the screen “ Academic status entry” form of a student module of university management system application
- Areas of the application to be tested

Test Scenario:

- When the office personals use this screen for the marks entry, calculate the status details, saving the information on student's basis and quit the form.

Test Procedure:

- The procedure for testing this screen is planned in such a way that the data entry, status calculation functionality, saving and quitting operations are tested in terms of Gui testing, Positive testing, Negative testing.

3.6. SYSTEM REQUIREMENTS

3.6.1. Software Requirements

Operating System	:	Windows XP/2003 or Linux/Solaris
User Interface	:	HTML, CSS
Client-side Scripting	:	Python
Programming Language	:	Python
Web Applications	:	JDBC, JNDI, Servlets, JSP
IDE/Workbench	:	Anaconda
Database	:	SQLite
Server Deployment	:	Tomcat6.0

3.6.2. Hardware Requirements

Processor	:	Pentium IV
Hard Disk	:	40GB
RAM	:	256MB

3.7 NON-FUNCTIONAL REQUIREMENTS:

Performance Requirements

Some Performance requirements identified is listed below:

- The database shall be able to accommodate a minimum of 10,000 records of students.
- The software shall support use of multiple users at a time. There are no other specific performance requirements that will affect development.

Security Requirements

Some of the factors that are identified to protect the software from accidental or malicious access, use, modification, destruction, or disclosure are described below. Specific requirements in this area could include the need to:

- Utilize certain cryptographic techniques
- Keep specific log or history data sets
- Assign certain functions to different modules
- Restrict communications between some areas of the program
- Check data integrity for critical variables
- Later version of the software will incorporate encryption techniques in the user/license authentication process.
- The software will include an error tracking log that will help the user understand what error occurred when the application crashed along with suggestions on how to prevent the error from occurring again.
- Communication needs to be restricted when the application is validating the user or license.(i.e., using https)

Portability Requirements

Some of the attributes of software that relate to the ease of porting the software to other host machines and/or operating systems.

This may include:

Java is used to develop the product. So it is easiest to port the software in any environment.

Maintainability

The user will be able to reset all options and all stored user variables to default settings.

Reliability

Some of the attributes identified for the reliability is listed below:

- All data storage for user variables will be committed to the database at the time of entry.
- Data corruption is prevented by applying the possible backup procedures and techniques.

Usability requirements

Some of the usability requirements identified for this system are listed below:

- A logical interface is essential to an easy to use system, speeding up common tasks.
- Error prevention is integral to the system and is provided in a number of formats from sanity hacks to limiting free-text input.

Availability

- All cached data will be rebuilt during every startup. There is no recovery of user data if it is lost. Default values of system data will be assigned when necessary

Software System Attributes

- There are a number of attributes of software that can serve as requirements. It is important that required attributes be specified so that their achievement can be objectively verified. The following items provide a partial list of examples.
- The input system will allow for inputting numbers, operands, special symbols and letter.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 SUMMARY

pygame differs from standard procedural programming. You've also learned how to:

- Implement event loops
- Draw items on the screen
- Play sound effects and music
- Handle user input

To do this, you used a subset of the pygame modules, including the display, mixer and music, time, image, event, and key modules. You also used several pygame classes, including Rect, Surface, Sound, and Sprite.

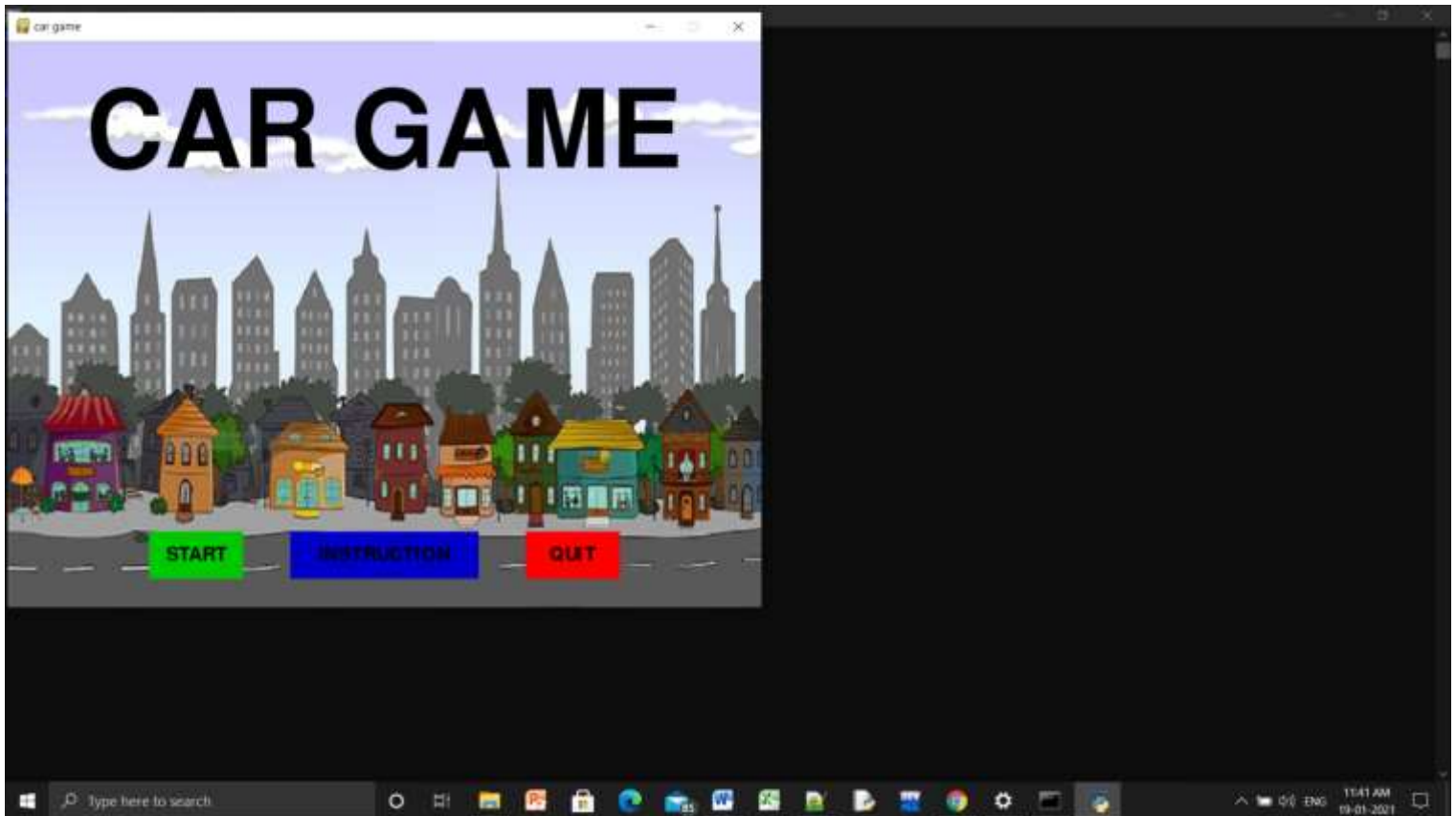
Multi core CPUs can be used easily. With dual core CPUs common, and 8 core CPUs cheaply available on desktop systems, making use of multi core CPUs allows you to do more in your game. Selected pygame functions release the dreaded python GIL, which is something you can do from C code.

Uses optimized C and Assembly code for core functions. C code is often 10-20 times faster than python code, and assembly code can easily be 100x or more times faster than python code.

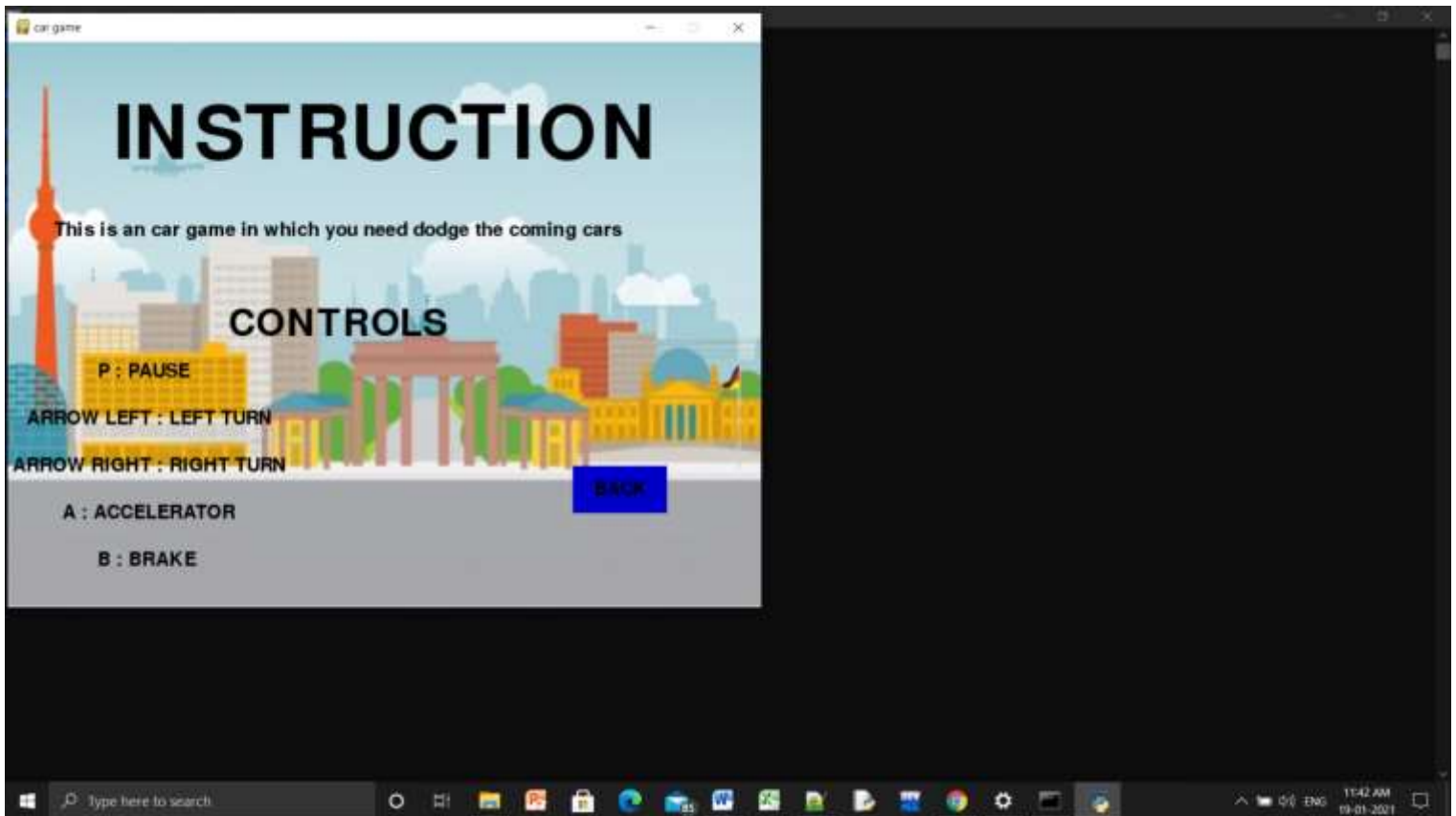
Comes with many operating systems. Just an apt-get, emerge, pkg_add, or yast install away. No need to mess with installing it outside of your operating system's package manager. Comes with binary pos system installers (and uninstallers) for Windows or MacOSX. Pygame does not require setup tools with even ctypes to install.

4.2 SCREEN SHOTS

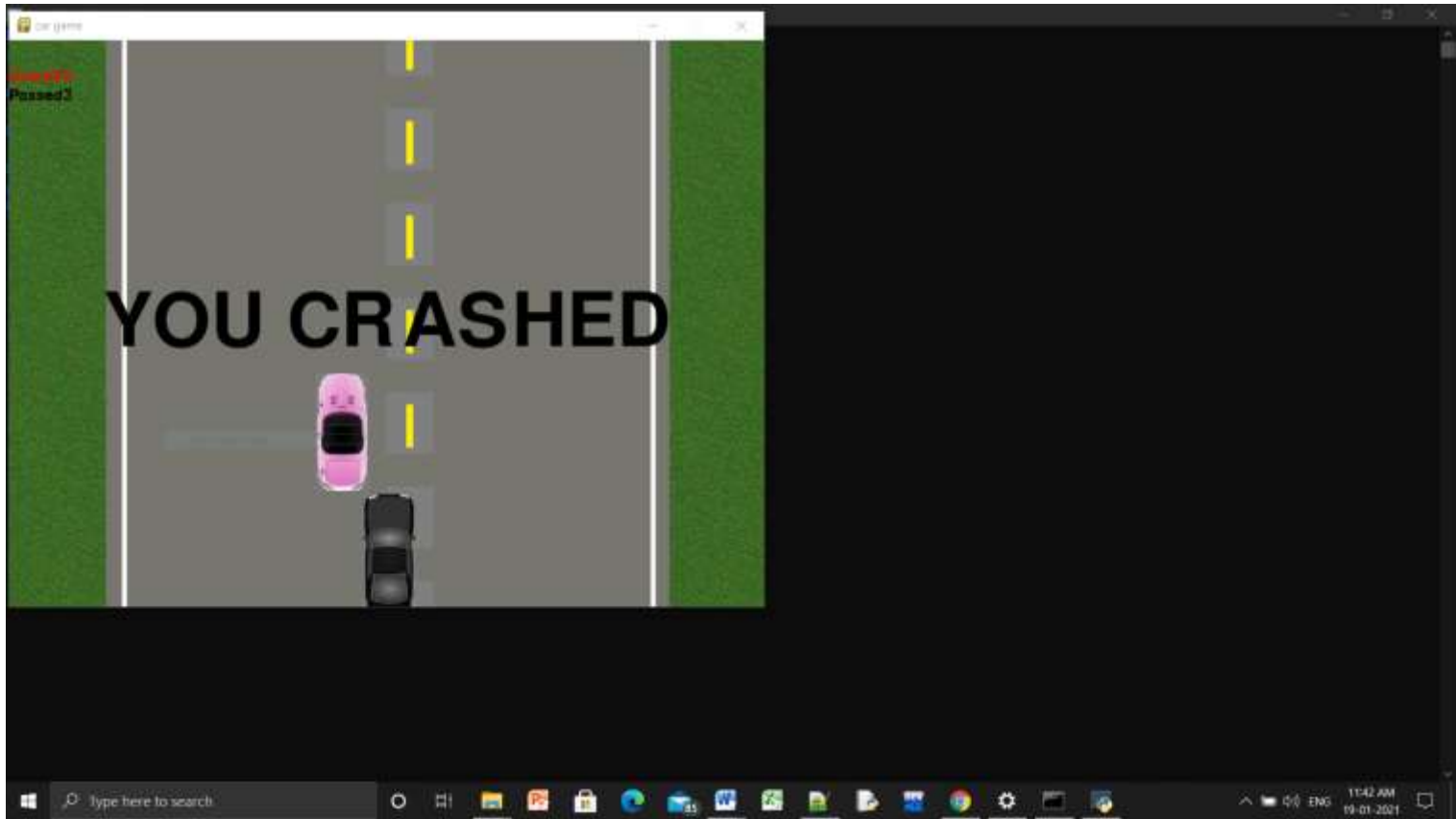
Page 1:



Page 2:



Page 3:



CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENT

5.1. CONCLUSION

pygame differs from standard procedural programming.

- Implement event loops
- Draw items on the screen
- Play sound effects and music
- Handle user input

To do this, you used a subset of the pygame modules, including the display, mixer and music, time, image, event, and key modules. You also used several pygame classes, including Rect, Surface, Sound, and Sprite.

5.2. FUTURE ENHANCEMENT

Data Science is a colossal pool of multiple data operations. These data operations also involve machine learning and statistics. Machine Learning algorithms are very much dependent on data. This data is fed to our model in the form of training set and test set which is eventually used for fine-tuning our model with various algorithmic parameters. By all means, advancement in Machine Learning is the key contributor towards the future of data science.

In particular, Data Science also covers:

- a. Data Integration.
- b. Distributed Architecture.
- c. Automating Machine learning.
- d. Data Visualisation.
- e. Dashboards and BI.
- f. Data Engineering.

- g. Deployment in production mode
- h. Automated, data-driven decisions.

5.3. REFERENCES :

- N. Schuessler and K. Axhausen, "Processing raw data from global positioning systems without additional information," *Transp. Res. Rec., J. Transp. Res. Board*, no. 2105, pp. 28–36, 2009.
- J. Wolf, "Using GPS data loggers to replace travel diaries in the collection of travel data," Georgia Inst. Technol., Citeseer, Tech. Rep., 2000.
- J. Ogle, R. Guensler, W. Bachman, M. Koutsak, and J. Wolf, "Accuracy of global positioning system for determining driver performance parameters," *Transp. Res. Rec., J. Transp. Res. Board*, no. 1818, pp. 12–24, 2002.

APPENDIX

6.1 SOURCE CODE

```
import pygame
pygame.init()
gray=(119,118,110)
black=(0,0,0)
red=(255,0,0)
green=(0,200,0)
blue=(0,0,200)
bright_red=(255,0,0)
bright_green=(0,255,0)
bright_blue=(0,0,255)
display_width=800
display_height=600
import time
import random

gamedisplays=pygame.display.set_mode((display_width,display_height))
pygame.display.set_caption("car game")
clock=pygame.time.Clock()
carimg=pygame.image.load('car1.jpg')
backgroundpic=pygame.image.load("download12.jpg")
yellow_strip=pygame.image.load("yellow strip.jpg")
strip=pygame.image.load("strip.jpg")
```

```

intro_background=pygame.image.load("background.jpg")
instruction_background=pygame.image.load("background2.jpg")
car_width=56
pause=False

def intro_loop():
    intro=True
    while intro:
        for event in pygame.event.get():
            if event.type==pygame.QUIT:
                pygame.quit()
                quit()
                sys.exit()
        gamedisplays.blit(intro_background,(0,0))
        largetext=pygame.font.Font('freesansbold.ttf',115)
        TextSurf,TextRect=text_objects("CAR GAME",largetext)
        TextRect.center=(400,100)
        gamedisplays.blit(TextSurf,TextRect)
        button("START",150,520,100,50,green,bright_green,"play")
        button("QUIT",550,520,100,50,red,bright_red,"quit")
        button("INSTRUCTION",300,520,200,50,blue,bright_blue,"intro")
        pygame.display.update()
        clock.tick(50)

def button(msg,x,y,w,h,ic,ac,action=None):
    mouse=pygame.mouse.get_pos()

```

```

click=pygame.mouse.get_pressed()
if x+w>mouse[0]>x and y+h>mouse[1]>y:
    pygame.draw.rect(gamedisplays,ac,(x,y,w,h))
    if click[0]==1 and action!=None:
        if action=="play":
            countdown()
        elif action=="quit":
            pygame.quit()
            quit()
            sys.exit()
        elif action=="intro":
            introduction()
        elif action=="menu":
            intro_loop()
        elif action=="pause":
            paused()
        elif action=="unpause":
            unpaused()

else:
    pygame.draw.rect(gamedisplays,ic,(x,y,w,h))
    smalltext=pygame.font.Font("freesansbold.ttf",20)
    textsurf,textrect=text_objects(msg,smalltext)
    textrect.center=((x+(w/2)),(y+(h/2)))
    gamedisplays.blit(textsurf,textrect)

```

```

def introduction():
    introduction=True
    while introduction:
        for event in pygame.event.get():
            if event.type==pygame.QUIT:
                pygame.quit()
                quit()
                sys.exit()
        gamedisplays.blit(instruction_background,(0,0))
        largertext=pygame.font.Font('freesansbold.ttf',80)
        smalltext=pygame.font.Font('freesansbold.ttf',20)
        mediumtext=pygame.font.Font('freesansbold.ttf',40)
        textSurf,textRect=text_objects("This is an car game in which you need dodge the
        coming cars",smalltext)
        textRect.center=((350),(200))
        TextSurf,TextRect=text_objects("INSTRUCTION",largertext)
        TextRect.center=((400),(100))
        gamedisplays.blit(TextSurf,TextRect)
        gamedisplays.blit(textSurf,textRect)
        stextSurf,stextRect=text_objects("ARROW LEFT : LEFT TURN",smalltext)
        stextRect.center=((150),(400))
        hTextSurf,hTextRect=text_objects("ARROW RIGHT : RIGHT TURN" ,smalltext)
        hTextRect.center=((150),(450))
        atextSurf,atextRect=text_objects("A : ACCELERATOR",smalltext)
        atextRect.center=((150),(500))
        rtextSurf,rtextRect=text_objects("B : BRAKE ",smalltext)
        rtextRect.center=((150),(550))

```

```

ptextSurf,ptextRect=text_objects("P : PAUSE ",smalltext)
ptextRect.center=((150),(350))
sTextSurf,sTextRect=text_objects("CONTROLS",mediumtext)
sTextRect.center=((350),(300))
gamedisplays.blit(sTextSurf,sTextRect)
gamedisplays.blit(stextSurf,stextRect)
gamedisplays.blit(hTextSurf,hTextRect)
gamedisplays.blit(atextSurf,atextRect)
gamedisplays.blit(rtextSurf,rtextRect)
gamedisplays.blit(ptextSurf,ptextRect)
button("BACK",600,450,100,50,blue,bright_blue,"menu")
pygame.display.update()
clock.tick(30)

```

```

def paused():

```

```

    global pause

```

```

    while pause:

```

```

        for event in pygame.event.get():

```

```

            if event.type==pygame.QUIT:

```

```

                pygame.quit()

```

```

                quit()

```

```

                sys.exit()

```

```

        gamedisplays.blit(instruction_background,(0,0))

```

```

        largetext=pygame.font.Font('freesansbold.ttf',115)

```

```

        TextSurf,TextRect=text_objects("PAUSED",largetext)

```

```

        TextRect.center=((display_width/2),(display_height/2))

```

```
gamedisplays.blit(TextSurf,TextRect)
button("CONTINUE",150,450,150,50,green,bright_green,"unpause")
button("RESTART",350,450,150,50,blue,bright_blue,"play")
button("MAIN MENU",550,450,200,50,red,bright_red,"menu")
pygame.display.update()
clock.tick(30)
```

```
def unpaused():
```

```
    global pause
```

```
    pause=False
```

```
def countdown_background():
```

```
    font=pygame.font.SysFont(None,25)
```

```
    x=(display_width*0.45)
```

```
    y=(display_height*0.8)
```

```
    gamedisplays.blit(backgroundpic,(0,0))
```

```
    gamedisplays.blit(backgroundpic,(0,200))
```

```
    gamedisplays.blit(backgroundpic,(0,400))
```

```
    gamedisplays.blit(backgroundpic,(700,0))
```

```
    gamedisplays.blit(backgroundpic,(700,200))
```

```
    gamedisplays.blit(backgroundpic,(700,400))
```

```
    gamedisplays.blit(yellow_strip,(400,100))
```

```
    gamedisplays.blit(yellow_strip,(400,200))
```

```
    gamedisplays.blit(yellow_strip,(400,300))
```

```
    gamedisplays.blit(yellow_strip,(400,400))
```

```
    gamedisplays.blit(yellow_strip,(400,100))
```

```

gamedisplays.blit(yellow_strip,(400,500))
gamedisplays.blit(yellow_strip,(400,0))
gamedisplays.blit(yellow_strip,(400,600))
gamedisplays.blit(strip,(120,200))
gamedisplays.blit(strip,(120,0))
gamedisplays.blit(strip,(120,100))
gamedisplays.blit(strip,(680,100))
gamedisplays.blit(strip,(680,0))
gamedisplays.blit(strip,(680,200))
gamedisplays.blit(carimg,(x,y))
text=font.render("DODGED: 0",True, black)
score=font.render("SCORE: 0",True,red)
gamedisplays.blit(text,(0,50))
gamedisplays.blit(score,(0,30))
button("PAUSE",650,0,150,50,blue,bright_blue,"pause")

```

```

def countdown():
    countdown=True

    while countdown:
        for event in pygame.event.get():
            if event.type==pygame.QUIT:
                pygame.quit()
                quit()
                sys.exit()
        gamedisplays.fill(gray)
        countdown_background()

```

```

largetext=pygame.font.Font('freesansbold.ttf',115)
TextSurf,TextRect=text_objects("3",largetext)
TextRect.center=((display_width/2),(display_height/2))
gamedisplays.blit(TextSurf,TextRect)
pygame.display.update()
clock.tick(1)
gamedisplays.fill(gray)
countdown_background()
largetext=pygame.font.Font('freesansbold.ttf',115)
TextSurf,TextRect=text_objects("2",largetext)
TextRect.center=((display_width/2),(display_height/2))
gamedisplays.blit(TextSurf,TextRect)
pygame.display.update()
clock.tick(1)
gamedisplays.fill(gray)
countdown_background()
largetext=pygame.font.Font('freesansbold.ttf',115)
TextSurf,TextRect=text_objects("1",largetext)
TextRect.center=((display_width/2),(display_height/2))
gamedisplays.blit(TextSurf,TextRect)
pygame.display.update()
clock.tick(1)
gamedisplays.fill(gray)
countdown_background()
largetext=pygame.font.Font('freesansbold.ttf',115)
TextSurf,TextRect=text_objects("GO!!!",largetext)
TextRect.center=((display_width/2),(display_height/2))

```

```
gamedisplays.blit(TextSurf,TextRect)
pygame.display.update()
clock.tick(1)
game_loop()
```

```
def obstacle(obs_startx,obs_starty,obs):
    if obs==0:
        obs_pic=pygame.image.load("car.jpg")
    elif obs==1:
        obs_pic=pygame.image.load("car1.jpg")
    elif obs==2:
        obs_pic=pygame.image.load("car2.jpg")
    elif obs==3:
        obs_pic=pygame.image.load("car4.jpg")
    elif obs==4:
        obs_pic=pygame.image.load("car5.jpg")
    elif obs==5:
        obs_pic=pygame.image.load("car6.jpg")
    elif obs==6:
        obs_pic=pygame.image.load("car7.jpg")
    gamedisplays.blit(obs_pic,(obs_startx,obs_starty))
```

```
def score_system(passed,score):
    font=pygame.font.SysFont(None,25)
    text=font.render("Passed"+str(passed),True,black)
    score=font.render("Score"+str(score),True,red)
    gamedisplays.blit(text,(0,50))
```

```
gamedisplays.blit(score,(0,30))
```

```
def text_objects(text,font):  
    textsurface=font.render(text,True,black)  
    return textsurface,textsurface.get_rect()
```

```
def message_display(text):  
    largertext=pygame.font.Font("freesansbold.ttf",80)  
    textsurf,textrect=text_objects(text,largertext)  
    textrect.center=((display_width/2),(display_height/2))  
    gamedisplays.blit(textsurf,textrect)  
    pygame.display.update()  
    time.sleep(3)  
    game_loop()
```

```
def crash():  
    message_display("YOU CRASHED")
```

```
def background():  
    gamedisplays.blit(backgroundpic,(0,0))  
    gamedisplays.blit(backgroundpic,(0,200))  
    gamedisplays.blit(backgroundpic,(0,400))  
    gamedisplays.blit(backgroundpic,(700,0))  
    gamedisplays.blit(backgroundpic,(700,200))
```

```
gamedisplays.blit(backgroundpic,(700,400))
gamedisplays.blit(yellow_strip,(400,0))
gamedisplays.blit(yellow_strip,(400,100))
gamedisplays.blit(yellow_strip,(400,200))
gamedisplays.blit(yellow_strip,(400,300))
gamedisplays.blit(yellow_strip,(400,400))
gamedisplays.blit(yellow_strip,(400,500))
gamedisplays.blit(strip,(120,0))
gamedisplays.blit(strip,(120,100))
gamedisplays.blit(strip,(120,200))
gamedisplays.blit(strip,(680,0))
gamedisplays.blit(strip,(680,100))
gamedisplays.blit(strip,(680,200))
```

```
def car(x,y):
    gamedisplays.blit(carimg,(x,y))
```

```
def game_loop():
    global pause
    x=(display_width*0.45)
    y=(display_height*0.8)
    x_change=0
    obstacle_speed=9
    obs=0
    y_change=0
    obs_startx=random.randrange(200,(display_width-200))
    obs_starty=-750
```

obs_width=56

obs_height=125

passed=0

level=0

score=0

y2=7

fps=120

bumped=False

while not bumped:

 for event in pygame.event.get():

 if event.type==pygame.QUIT:

 pygame.quit()

 quit()

 if event.type==pygame.KEYDOWN:

 if event.key==pygame.K_LEFT:

 x_change=-5

 if event.key==pygame.K_RIGHT:

 x_change=5

 if event.key==pygame.K_a:

 obstacle_speed+=2

 if event.key==pygame.K_b:

 obstacle_speed-=2

 if event.type==pygame.KEYUP:

 if event.key==pygame.K_LEFT or event.key==pygame.K_RIGHT:

 x_change=0

```
x+=x_change
```

```
pause=True
```

```
gamedisplays.fill(gray)
```

```
rel_y=y2%backgroundpic.get_rect().width
```

```
gamedisplays.blit(backgroundpic,(0,rel_y-backgroundpic.get_rect().width))
```

```
gamedisplays.blit(backgroundpic,(700,rel_y-backgroundpic.get_rect().width))
```

```
if rel_y<800:
```

```
    gamedisplays.blit(backgroundpic,(0,rel_y))
```

```
    gamedisplays.blit(backgroundpic,(700,rel_y))
```

```
    gamedisplays.blit(yellow_strip,(400,rel_y))
```

```
    gamedisplays.blit(yellow_strip,(400,rel_y+100))
```

```
    gamedisplays.blit(yellow_strip,(400,rel_y+200))
```

```
    gamedisplays.blit(yellow_strip,(400,rel_y+300))
```

```
    gamedisplays.blit(yellow_strip,(400,rel_y+400))
```

```
    gamedisplays.blit(yellow_strip,(400,rel_y+500))
```

```
    gamedisplays.blit(yellow_strip,(400,rel_y-100))
```

```
    gamedisplays.blit(strip,(120,rel_y-200))
```

```
    gamedisplays.blit(strip,(120,rel_y+20))
```

```
    gamedisplays.blit(strip,(120,rel_y+30))
```

```
    gamedisplays.blit(strip,(680,rel_y-100))
```

```
    gamedisplays.blit(strip,(680,rel_y+20))
```

```
    gamedisplays.blit(strip,(680,rel_y+30))
```

```
y2+=obstacle_speed
```

```

obs_starty-=(obstacle_speed/4)
obstacle(obs_startx,obs_starty,obs)
obs_starty+=obstacle_speed
car(x,y)
score_system(passed,score)
if x>690-car_width or x<110:
    crash()
if x>display_width-(car_width+110) or x<110:
    crash()
if obs_starty>display_height:
    obs_starty=0-obs_height
    obs_startx=random.randrange(170,(display_width-170))
    obs=random.randrange(0,7)
    passed=passed+1
    score=passed*10
    if int(passed)%10==0:
        level=level+1
        obstacle_speed+2
        largetext=pygame.font.Font("freesansbold.ttf",80)
        textsurf,textrect=text_objects("LEVEL"+str(level),largetext)
        textrect.center=((display_width/2),(display_height/2))
        gamedisplays.blit(textsurf,textrect)
        pygame.display.update()
        time.sleep(3)

```

```
    if y < obs_starty + obs_height:
        if x > obs_startx and x < obs_startx + obs_width or x + car_width > obs_startx
        and x + car_width < obs_startx + obs_width:
            crash()
            button("Pause", 650, 0, 150, 50, blue, bright_blue, "pause")
            pygame.display.update()
            clock.tick(60)
intro_loop()
game_loop()
pygame.quit()
quit()
```