Project Report on

Flappy Bird

2020-2021

Yash Baidwan

Class	-	12 (Commerce)
Roll Number	-	13657786

Acknowledgement

I would like to express my special thanks of gratitude to my teacher Ms. Prabhjit Kaur as well as our principal Ms. Manbir Brar who gave me the golden opportunity to do this wonderful project on the topic Python, which also helped me in doing a lot of Research and I came to know about many new things I am really thankful to them.

Secondly I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

Jasdish Baidwan

Certificate By Guide

This is to certify that Jasdish Baidwan of XII (Commerce) has completed his project file in partial fulfillment of the requirements set by the CBSE. It is authentic work carried out by him under my supervision. He has taken proper care and shown utmost sincerity in completion of the project.

I certify that this project is upto my expectations and as per the guidelines issued by the CBSE.

Ms. Prabhjit Kaur

(PGT Computer Science)

INDEX

Analysis

System Requirements

1. HARDWARE:

Minimum:

Intel Pentium Dual Core or AMD equivalent

1 GB RAM

1 GB empty storage

Recommended:

Intel Core i3 (5th Gen) or AMD equivalent

4 GB RAM

1 GB empty storage

2. SOFTWARE:

Minimum:

Windows 7 (32-bit) / Linux

Python 3.7 (with pygame library installed)

Recommended:

Windows 10 (64-bit) / Linux Deb.

Python 3.7 (with pygame library installed)

Feasibility Study

Feasibility is defined as the practical extent to which a project can be performed successfully. To evaluate feasibility, a feasibility study is performed, which determines whether the solution considered to accomplish the requirements is practical and workable in the software. Information such as resource availability, cost estimation for software development, benefits of the software to the organization after it is developed and cost to be incurred on its maintenance are considered during the feasibility study. The objective of the feasibility study is to establish the reasons for developing the software that is acceptable to users, adaptable to change and conformable to established standards. Various other objectives of feasibility study are listed below.

- To analyze whether the software will meet organizational requirements.

- To determine whether the software can be implemented using the current technology and within the specified budget and schedule.

~

A feasibility study is an assessment of the practicality of a proposed project or system. A feasibility study aims to objectively and rationally uncover the strengths and weaknesses of an existing business or proposed venture, opportunities and threats present in the natural environment, the resources required to carry through, and ultimately the prospects for success. In its simplest terms, the two criteria to judge feasibility are cost required and value to be attained.

~

Types of Feasibility

• Economical Feasibility

Economic feasibility is a kind of cost-benefit analysis of the examined project, which assesses whether it is possible to implement it. This term means the assessment and analysis of a project's potential to support the decision-making process by objectively and rationally identifying its strengths, weaknesses, opportunities and risks associated with it, the resources that will be needed to implement the project, and an assessment of its chances of success. It consists of market analysis, economic analysis, technical and strategic analysis.

~

Economics analysis is the most frequent use method for evaluating the effectiveness of the candidates the benefits and savings that are expected from system and compare them with cost.Though most working people don’t give much importance to them, video games are a good stress buster which in turn helps everyone be more productive therefore making it economically feasible.

~

• Technical Feasibility

This assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the technical team is capable of converting the ideas into working systems. Technical feasibility also involves the evaluation of the hardware, software, and other technical requirements of the proposed system. As an exaggerated example, an organization wouldn't want to try to put Star Trek's transporters in their building—currently, this project is not technically feasible.

~

Technical feasibility center on the existing computer system and to what extent it can support the proposed task. This involves financial consideration to accommodate technical enhancements.

It is technically feasible because whatever technology is needed to develop this software is easily available.

~

Overview

On February 10th, 2014, the creator of Flappy Bird, a popular side scrolling game for mobile devices, removed the game from mobile application stores (Apple, Android, etc.). The reason for the removal was due to the game becoming “an addictive product” that had “become a problem” (Nyugen). Currently Flappy Bird is only available to those who downloaded the game before its deletion from application stores or through a fan created website, flappybird.io.

This final project aims to implement this popular and exciting game in hardware. Previous implementations of this game, such as the original mobile app or fancreated online version, exist in software form. As a result, bringing this game into the hardware realm truly distinguishes our project from previous implementations of Flappy Bird. In the preexisting software implemented versions of this game, a small bird must hop to avoid obstacles in the form of green pipes that scroll across the screen from right to left. The user taps the screen or clicks a button to make the bird jump up, but otherwise the bird is constantly falling. Due to the counter intuitive control scheme, which required the user to remain vigilant, constantly monitoring the bird’s path of movement during the game, Flappy Bird became renowned for its difficulty and infuriating controls.

Therefore, a large change in the player’s vertical position, as monitored by the vision tracking component, will trigger a hop for the player’s on screen character. Additionally, during game play, the player’s face will be layered onto the bird sprite, allowing for a truly personalized experience. **The ideal outcome would be to deliver an aesthetically pleasing and physically engaging game that demonstrates successful implementation of the game integrated with the vision tracking component.**

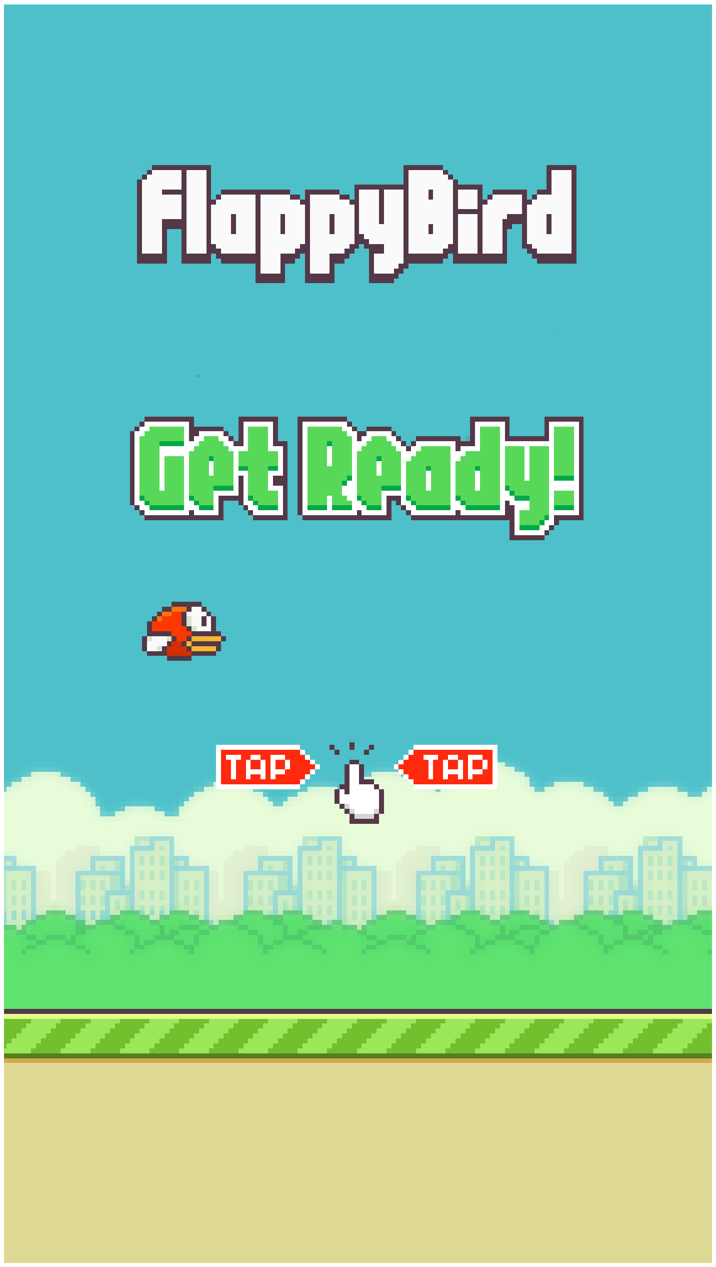
Code

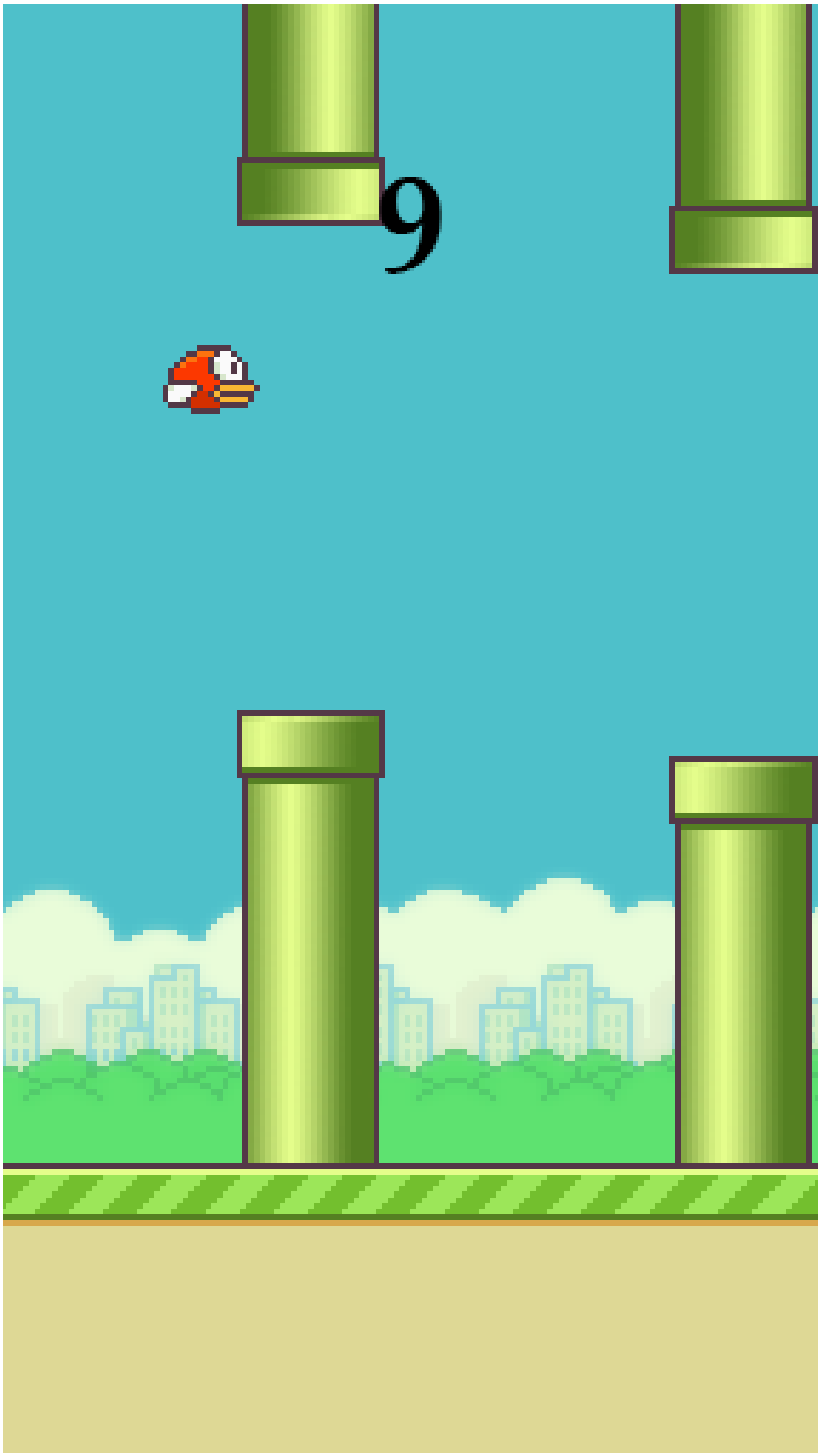
```
import random # For generating random numbers
import sys # We will use sys.exit to exit the program
import pygame
from pygame.locals import * # Basic pygame imports

# Global Variables for the game

FPS = 32
SCREENWIDTH = 289
SCREENHEIGHT = 511
SCREEN = pygame.display.set_mode((SCREENWIDTH, SCREENHEIGHT))
GROUNDY = SCREENHEIGHT * 0.8
GAME_SPRITES = {}
GAME_SOUNDS = {}
PLAYER = 'gallery/sprites/bird.png'
BACKGROUND = 'gallery/sprites/background.png'
PIPE = 'gallery/sprites/pipe.png'

def welcomeScreen():

    """
    Shows welcome images on the screen
    """

    playerx = int(SCREENWIDTH/5)
    playery = int((SCREENHEIGHT - GAME_SPRITES['player'].get_height())/2)
    messagex = int((SCREENWIDTH - GAME_SPRITES['message'].get_width())/2)
    messagey = int(SCREENHEIGHT*0.13)
    basex = 0
    while True:
        for event in pygame.event.get():
            # if user clicks on cross button, close the game
            if event.type == QUIT or (event.type==KEYDOWN and event.key
            == K_ESCAPE):
                pygame.quit()
                sys.exit()

            # If the user presses space or up key, start the game
            elif event.type==KEYDOWN and (event.key==K_SPACE or event.key ==
            K_UP):
                return
            else:
                SCREEN.blit(GAME_SPRITES['background'], (0, 0))
                SCREEN.blit(GAME_SPRITES['player'], (playerx,playery))
                SCREEN.blit(GAME_SPRITES['message'],(messagex,messagey)) ))
                SCREEN.blit(GAME_SPRITES['base'], (basex, GROUNDY))
                pygame.display.update()
                FPSCLOCK.tick(FPS)

def
    mainGame():score = 0
    playerx = int(SCREENWIDTH/5)
    playery = int(SCREENWIDTH/2)
    basex = 0

    # Create 2 pipes for blitting on the screen
    newPipe1 = getRandomPipe()
    newPipe2 = getRandomPipe()

    # my List of upper pipes
    upperPipes = [
        {'x': SCREENWIDTH+200, 'y':newPipe1[0]['y']},
        {'x': SCREENWIDTH+200+(SCREENWIDTH/2), 'y':newPipe2[0]['y']},
    ]

    # my List of lower pipes
    lowerPipes = [
        {'x': SCREENWIDTH+200, 'y':newPipe1[1]['y']},
        {'x': SCREENWIDTH+200+(SCREENWIDTH/2), 'y':newPipe2[1]['y']}
    ]

    pipeVelX = -4

    playerVelY = -9
    playerMaxVelY = 10
    playerMinVelY = -8
    playerAccY = 1

    playerFlapAccv = -8 # velocity while flapping
    playerFlapped = False # It is true only when the bird is flapping


    while True:
        for event in pygame.event.get():
            if event.type == QUIT or (event.type == KEYDOWN and event.key ==
            K_ESCAPE):
                pygame.quit()
                sys.exit()
            if event.type == KEYDOWN and (event.key == K_SPACE or event.key ==
            K_UP):
                if playery > 0:
              playerVelY = playerFlapAccv
              playerFlapped = True
              GAME_SOUNDS['wing'].play()


crashTest = isCollide(playerx, playery, upperPipes, lowerPipes)
# This function will return true if the player is crashed
```

```
if crashTest:
            return

        #check for score
        playerMidPos = playerx + GAME_SPRITES['player'].get_width()/2
        for pipe in upperPipes:
            pipeMidPos = pipe['x'] + GAME_SPRITES['pipe'][0].get_width()/2
            if pipeMidPos<= playerMidPos < pipeMidPos +4:
                score +=1
                print(f"Your score is {score}")
                GAME_SOUNDS['point'].play()


        if playerVelY <playerMaxVelY and not playerFlapped:
            playerVelY += playerAccY

        if playerFlapped:
            playerFlapped = False
        playerHeight = GAME_SPRITES['player'].get_height()
        playery = playery + min(playerVelY, GROUNDY - playery - playerHeight)

        # move pipes to the left
        for upperPipe , lowerPipe in zip(upperPipes, lowerPipes):
            upperPipe['x'] += pipeVelX
            lowerPipe['x'] += pipeVelX

        # Add a new pipe when the first is about to cross the leftmost part of
        the screen
        if 0<upperPipes[0]['x']<5:
            newpipe = getRandomPipe()
            upperPipes.append(newpipe[0])
            lowerPipes.append(newpipe[1])

        # if the pipe is out of the screen, remove it
        if upperPipes[0]['x'] < -GAME_SPRITES['pipe'][0].get_width():
            upperPipes.pop(0)
            lowerPipes.pop(0)

        # Lets blit our sprites now
        SCREEN.blit(GAME_SPRITES['background'], (0, 0))
        for upperPipe, lowerPipe in zip(upperPipes, lowerPipes):
            SCREEN.blit(GAME_SPRITES['pipe'][0], (upperPipe['x'],
            upperPipe['y']))
            SCREEN.blit(GAME_SPRITES['pipe'][1], (lowerPipe['x'],
            lowerPipe['y']))

        SCREEN.blit(GAME_SPRITES['base'], (basex, GROUNDY))
        SCREEN.blit(GAME_SPRITES['player'], (playerx, playery))
        myDigits = [int(x) for x in list(str(score))]
        width = 0
        for digit in myDigits:
            width += GAME_SPRITES['numbers'][digit].get_width()
        Xoffset = (SCREENWIDTH - width)/2
        for digit in myDigits:
            SCREEN.blit(GAME_SPRITES['numbers'][digit], (Xoffset,SCREENHEIGHT*
            0.12))
            Xoffset += GAME_SPRITES['numbers'][digit].get_width()
        pygame.display.update()
        FPSCLOCK.tick(FPS)

def isCollide(playerx, playery, upperPipes, lowerPipes):
    if playery> GROUNDY - 25  or playery<0:
        GAME_SOUNDS['hit'].play()
        return True

    for pipe in upperPipes:
        pipeHeight = GAME_SPRITES['pipe'][0].get_height()
        if(playery < pipeHeight + pipe['y'] and abs(playerx - pipe['x']) <
        GAME_SPRITES['pipe'][0].get_width()):
            GAME_SOUNDS['hit'].play()
            return True

    for pipe in lowerPipes:
        if (playery + GAME_SPRITES['player'].get_height() > pipe['y']) and
        abs(playerx - pipe['x']) < GAME_SPRITES['pipe'][0].get_width():
            GAME_SOUNDS['hit'].play()
            return True

    return False

def getRandomPipe():
    """
    Generate positions of two pipes(one bottom straight and one top rotated )
    for blitting on the screen
    """
    pipeHeight = GAME_SPRITES['pipe'][0].get_height()
    offset = SCREENHEIGHT/3
    y2 = offset + random.randrange(0, int(SCREENHEIGHT -
    GAME_SPRITES['base'].get_height()  - 1.2 *offset))
    pipeX = SCREENWIDTH + 10
    y1 = pipeHeight - y2 + offset
    pipe = [
        {'x': pipeX, 'y': -y1}, #upper Pipe
        {'x': pipeX, 'y': y2} #lower Pipe
    ]
    return pipe
```

```
if __name__ == "__main__":
    # This will be the main point from where our game will start
    pygame.init() # Initialize all pygame's modules
    FPSCLOCK = pygame.time.Clock()
    pygame.display.set_caption('Flappy Bird by Jasdish')
            GAME_SPRITES['numbers'] = (
          pygame.image.load('gallery/sprites/0.png').convert_alpha(),
        pygame.image.load('gallery/sprites/1.png').convert_alpha(),
        pygame.image.load('gallery/sprites/2.png').convert_alpha(),
        pygame.image.load('gallery/sprites/3.png').convert_alpha(),
        pygame.image.load('gallery/sprites/4.png').convert_alpha(),
        pygame.image.load('gallery/sprites/5.png').convert_alpha(),
        pygame.image.load('gallery/sprites/6.png').convert_alpha(),
        pygame.image.load('gallery/sprites/7.png').convert_alpha(),
        pygame.image.load('gallery/sprites/8.png').convert_alpha(),
        pygame.image.load('gallery/sprites/9.png').convert_alpha(),
    )

    GAME_SPRITES['message']
    =pygame.image.load('gallery/sprites/message.png').convert_alpha()
    GAME_SPRITES['base']
    =pygame.image.load('gallery/sprites/base.png').convert_alpha()
    GAME_SPRITES['pipe'] =(pygame.transform.rotate(pygame.image.load(
    PIPE).convert_alpha(), 180),
    pygame.image.load(PIPE).convert_alpha()
    )

    # Game sounds
    GAME_SOUNDS['die'] = pygame.mixer.Sound('gallery/audio/die.wav')
    GAME_SOUNDS['hit'] = pygame.mixer.Sound('gallery/audio/hit.wav')
    GAME_SOUNDS['point'] = pygame.mixer.Sound('gallery/audio/point.wav')
    GAME_SOUNDS['swoosh'] = pygame.mixer.Sound('gallery/audio/swoosh.wav')
    GAME_SOUNDS['wing'] = pygame.mixer.Sound('gallery/audio/wing.wav')
    GAME_SPRITES['background'] = pygame.image.load(BACKGROUND).convert()
    GAME_SPRITES['player'] = pygame.image.load(PLAYER).convert_alpha()

    while True:
        welcomeScreen() # Shows welcome screen to the user until he presses a
        button
        mainGame() # This is the main game function
```

Output




9

Testing

• Alpha Testing

It is the phase of game testing where the game is still in the development phase along with which parallel testing is done to ensure that the game is developed without any glitches and is working smoothly without crashing. This stage requires intense documentation of hardware and software failure so that the game developers can fix it at earlier stages.

As a game tester, one is not going to have fun during this phase as the some of the major game functionality and Graphical UI may still be under development phase. However, a tester who loves to solve the game problem and tries to map the solution for the betterment of the game will surely enjoy this phase. It is expected that the game tester provides some good piece of suggestions for further game development.

~

It is the most common type of testing used in the software industry. The objective of this testing is to identify all possible issues or defects before releasing it into the market or to the user. It is conducted at the developer's site.

~

• Beta Testing

During beta testing, the game is almost production ready with all the major issues being fixed. In this phase, the game testers are required to extensively find all the possible ways to break the game along the lookout for all minor issues.

During Beta Testing, the game needs to pass through many testing methodologies such as performance testing, stress testing, and game compliance testing. Tester runs the game seamlessly for hours together on many devices to check for any performance or server downfall.

~

It is a formal type of software testing which is carried out by the customers. It is performed in a real environment before releasing the products into the market for the actual end-users. It is carried out to ensure that there are no major failures in the software or product and it satisfies the business requirement. Beta Testing is successful when the customer accepts the software.

~

• White **Box Testing**

In contrast to black box testing, white box testing gives the tester an opportunity to exercise the source code directly in ways no end user ever could. It can be a daunting challenge for a white box tester to read a piece of game code and predict every single manner in which it will interact with every other bit of code, and whether the code has accounted for every combination and order of inputs possible. Testing a game using only white box methods is also extremely difficult because it is nearly impossible to account for the complexity of the player feedback loop. There are, however, situations in which white box testing is more practical and necessary than black box testing.

~

White box testing is based on the knowledge about the internal logic of an application's code. It is also known as Glass box Testing. Internal Software and code working should be known for performing this type of testing. These tests are based on the coverage of the code statements, branches, paths, conditions etc.

~

• Black Box Testing

Black box testing involves testing a system with no prior knowledge of its internal workings. A tester provides an input, and observes the output generated by the system under test. This makes it possible to identify how the system responds to expected and unexpected user actions, its response time, usability issues and reliability issues.

Black box testing is a powerful testing technique because it exercises a system end-to-end. Just like end-users “don’t care” how a system is coded or architected, and expect to receive an appropriate response to their requests, a tester can simulate user activity and see if the system delivers on its promises. Along the way, a black box test evaluates all relevant subsystems, including UI/UX, web server or application server, database, dependencies, and integrated systems.

~

It is a software testing, method in which the internal structure or design of the item to be tested is not known to the tester. This method of testing can be applied virtually to every level of the software testing.

~

Maintenance

Program maintenance is the process of modifying a software or program after delivery to achieve any of these outcomes −

- Correct errors
- Improve performance
- Add functionalities
- Remove obsolete portions

Despite the common perception that maintenance is required to fix errors that come up after the software goes live, in reality most of the maintenance work involves adding minor or major capabilities to existing modules. For example, some new data is added to a report, a new field added to entry forms, code to be modified to incorporate changed government laws, etc.

~

Programming maintenance refers to the modifications in the program. After it has been completed, in order to meet changing requirement or to take care of the errors that shown up.

~

• Corrective Maintenance

Corrective maintenance is the category of maintenance tasks that are performed to rectify and repair faulty systems and equipment. The purpose of corrective maintenance is to restore systems that have broken down. Corrective maintenance can be synonymous with breakdown or reactive maintenance.

Corrective maintenance orders are typically initiated when an additional problem is discovered during a separate work order. For example, if a maintenance technician spots an issue during an emergency repair, as part of a routine inspection or in the process of conducting preventive maintenance, that issue can turn into a corrective maintenance order. That corrective maintenance task is then planned and scheduled for a future time. During the execution of corrective maintenance work, the asset is repaired, restored, or replaced as needed.

~

When the program after compilation shows error because of some unexpected situations, untested areas such errors are fixed up by Corrective maintenance.

~

• Adaptive Maintenance

Adaptive maintenance refers to the enforcement of changes in the monitoring, use or other operational details of a metallic structure or object to prevent corrosion from spreading from one part of the metal where it is already present to another.

Typically, adaptive maintenance involves upgrading corrosion monitoring software systems to adjust the parameters the software recognizes as corrosion-inducing properties. The better the software is at reading such properties, the better the degree of corrosion prevention in a given application.

Adaptive maintenance consists of changing, often self-regulating software that monitors changes in an external environment. The term environment in this context refers to the conditions that influence corrosion (such as moisture, chemicals, oxygen, etc.).

~

Changes in the environment in which an information system operates may lead to system management. To accommodate changing needs time to time maintenance is done and is called Adaptive maintenance.

~

• Preventive Maintenance

Preventive maintenance (or preventative maintenance) is work that is performed regularly (on a scheduled basis) in order to minimize the chance that a certain piece of equipment will fail and cause costly unscheduled downtime. Preventative maintenance is hence performed while the equipment is still in working condition.

Preventive maintenance (or preventative maintenance) is maintenance that is regularly performed on a piece of equipment to lessen the likelihood of it failing. It is performed while the equipment is still working so that it does not break down unexpectedly. In terms of the complexity of this maintenance strategy, it falls between reactive (or run-to-failure) maintenance and predictive maintenance.

~

If possible the errors could be anticipated before they actually occur; the maintenance is called Preventive maintenance.

~

• Perfective Maintenance

Perfective maintenance mainly deals with implementing new or changed user requirements. Perfective maintenance involves making functional enhancements to the system in addition to the activities to increase the system's performance even when the changes have not been suggested by faults. This includes enhancing both the function and efficiency of the code and changing the functionalities of the system as per the users' changing needs.

Examples of perfective maintenance include modifying the payroll program to incorporate a new union settlement and adding a new report in the sales analysis system. Perfective maintenance accounts for 50%, that is, the largest of all the maintenance activities.

~

In this rapidly changing world, information technology is the fastest growing area. If te existing system is maintained to keep tuned with the new features, new facilities, new capabilities, it is said to be Perfective maintenance.

~

Appendix

Module:

pygame:

Package for 2d game development

Functions:

init():	Starts up pygame
surface():	Creates a blank canvas to draw objects on
blit():	Shows pixels on screen
mixer():	Used for adding sounds and music
event.get():	To check if a key or button is pressed
transform.scale2x():	Scales a surface to twice it's size
draw.circle():	Draws a circle on the screen
draw.rect():	Draws a rectangle on the screen
image.load():	Loads an image which can be drawn with blit() later
font.render():	To save text as image which can be later drawn to a surface

Conclusion

Implementing this game in hardware was an enjoyable project for all of us. It paired a fun final product with a challenging implementation, an ideal combination for a project in this class. The work was modular and divided well between three people, so we could optimize the division of labor. Our goals were achievable, and we accomplished what we wanted. Overall, this project helped all of us gain experience working with hardware implementation and interfacing with the outside world.

Bibliography

Quora

Pygame

Limble cmms

Informatics Practices by Sumita Arora