

COMPUTER SCIENCE YEAR 9

**PERTH MODERN SCHOOL**

**Walter Wilson**

**5/24/2022**

ALFIMOV Fedor

Software Development

**Assessment Task for Year 9 Course**

**TYPE:** Production

**OUTCOMES:** Outcome 1:Technology process; Outcome 2: Knowledge and understanding of computer-based systems; Outcome

**CONTENT:** Programming

## TASK 2: Programming Production (52 marks) (25%)

In this task you are required to design, plan, develop and test a new and original application with a very clear purpose; educational, entertainment, informative, business or any other idea you may have.

Be creative!

Time allocation: 2 Weeks

1. Initial Concepts (13 marks)

* Give a detail description of your initial concepts; description of your idea, name, purpose, type of App,

Initial concepts:

Similar to Minecraft in the way that you can place blocks.

Meant to be a mix between a platformer and Minecraft – less of a stress on either sides, as you can create your own ‘platformer levels’, rather than the standard prebuilt platformer, or the infinite world of Minecraft.

Name: Crine Maft

A play on Minecraft by swapping first letters.

Purpose: Entertainment, it is just a time killer for someone to enjoy for 5 minutes when they have a break.

Type of App: Python Pygame application. This means that it is a GUI application, meaning that it has proper visuals, including things like rendering and dechunking.

* Analyse similar apps available - 3 samples

Platformer\_project – dafluffypotato

Style

Has a very messy codebase. Everything is inside one for loop, drawing, movement and collisions. Makes the code very hard to read but more optimal.

More lenient towards the Minecraft aspect by having infinite world generation through chunking, but it is impossible to place blocks.

Has an interesting block storage system through txt files and changing them. I was inspired by this idea, and put my own spin on it, creating my own world and text reader and implementation of placing blocks and editing the txt.

Performance

Good performance because it is a small game, caps out at 60 fps because of the

Very similar in game mechanics.

Aeroblaster – DafluffyPotato

Style

Much Neater Source Code – it is split across many various subfiles, which is another thing that I took inspiration of.

It is much easier to create your own codebase in python because all it takes to import another file is to place

import file

at the top of the python code, making source code much more readable.

Style

Pixel art visual style similar to platformer\_project. Very simple due to the limitations of vector graphics in pygame.

Performance

High performance due to it being a small game

Paper Minecraft – MIT course

Style

Visual style including vector graphics

Nicely formatted source code over multiple files.

Great animal AI created in 2d using modified A\* pathfinding.

I was going to implement animals on the basis of this modified pathfinding, but did not have enough time to create

Performance

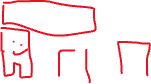
Medium performance – up to 100 fps on a high end machine but around 30 on a low end machine.

* Resources; programs/programming language, equipment, skills and expertise of the developer.
* Python and pygame are used.
* I am the developer who has skills and expertise in python with over 500 hours developing code.

1. Plan and Design (13 marks)

* Create screen layouts, sketches, colour scheme, graphics, user interface, etc.

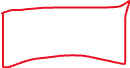
Main Game



Pause Menu



Main Menu



Images:

Background pattern

Description automatically generated

* Create a flowchart/algorithm describing the program

Main Program:

IMPORT pygame, sys

IMPORT numpy as np

IMPORT escapeMenu

SET Currblock TO '1'

SET clock TO pygame.time.Clock()

player\_x\_block=0

player\_y\_block=0

from pygame.locals IMPORT \*

pygame.init() # initiates pygame

pygame.display.set\_caption('CRINEMAFT')

SET WINDOW\_SIZE TO (600,400)

SET screen TO pygame.display.set\_mode(WINDOW\_SIZE,0,32) # initiate the window

SET display TO pygame.Surface((300,200)) # used as the surface FOR rendering, which is scaled

SET moving\_right TO False

SET moving\_left TO False

SET vertical\_momentum TO 0

SET air\_timer TO 0

SET true\_scroll TO [0,0]

DEFINE FUNCTION load\_map(path):

SET f TO open(path + '.txt','r')

SET data TO f.read()

f.close()

SET data TO data.split('\n')

SET game\_map TO []

FOR row IN data:

game\_map.append(list(row))

RETURN game\_map

SET game\_map TO load\_map('map')

SET grass\_img TO pygame.image.load('Images/grass.png')

SET dirt\_img TO pygame.image.load('Images/dirt.png')

SET flower\_img TO pygame.image.load('Images/flower.png')

flower\_img.set\_colorkey((0,0,0))

SET planks\_img TO pygame.image.load('Images/planks.png')

SET spruceplanks\_img TO pygame.image.load('Images/spruceplank.png')

SET spruce\_wood\_img TO pygame.image.load('Images/sprucewood.png')

SET cobble\_img TO pygame.image.load('Images/cobble.png')

SET leaves\_png TO pygame.image.load('Images/leaves.png')

leaves\_png.set\_colorkey((0,0,0))

SET planks\_img TO pygame.image.load('Images/planks.png')

SET stone\_img TO pygame.image.load('Images/stone.png')

SET wood\_img TO pygame.image.load('Images/wood.png')

SET player\_img TO pygame.image.load('Images/player.png').convert()

player\_img.set\_colorkey((255,255,255))

SET player\_rect TO pygame.Rect(100,100,5,13)

SET background\_objects TO [[0.25,[120,10,70,400]],[0.25,[280,30,40,400]],[0.5,[30,40,40,400]],[0.5,[130,90,100,400]],[0.5,[300,80,120,400]]]

DEFINE FUNCTION collision\_test(rect,tiles):

SET hit\_list TO []

FOR tile IN tiles:

IF rect.colliderect(tile):

hit\_list.append(tile)

RETURN hit\_list

DEFINE FUNCTION move(rect,movement,tiles):

SET collision\_types TO {'top':False,'bottom':False,'right':False,'left':False}

rect.x += movement[0]

SET hit\_list TO collision\_test(rect,tiles)

FOR tile IN hit\_list:

IF movement[0] > 0:

SET rect.right TO tile.left

SET collision\_types['right'] TO True

ELSEIF movement[0] < 0:

SET rect.left TO tile.right

SET collision\_types['left'] TO True

rect.y += movement[1]

SET hit\_list TO collision\_test(rect,tiles)

FOR tile IN hit\_list:

IF movement[1] > 0:

SET rect.bottom TO tile.top

SET collision\_types['bottom'] TO True

ELSEIF movement[1] < 0:

SET rect.top TO tile.bottom

SET collision\_types['top'] TO True

RETURN rect, collision\_types

DEFINE FUNCTION Main():

# Globals

global player\_x\_block

global player\_y\_block

global Currblock

global true\_scroll

global air\_timer

global moving\_right

global moving\_left

global vertical\_momentum

global player\_rect

global game\_map

global background\_objects

global screen

global display

global grass\_img

global dirt\_img

global flower\_img

global planks\_img

global spruceplanks\_img

global spruce\_wood\_img

global cobble\_img

global leaves\_png

global stone\_img

global wood\_img

global player\_img

# game loop

WHILE True:

display.fill((146,244,255)) # clear screen by filling it with blue

true\_scroll[0] += (player\_rect.x-true\_scroll[0]-152)/20

true\_scroll[1] += (player\_rect.y-true\_scroll[1]-106)/20

SET scroll TO true\_scroll.copy()

SET scroll[0] TO int(scroll[0])

SET scroll[1] TO int(scroll[1])

pygame.draw.rect(display,(7,80,75),pygame.Rect(0,120,300,80))

FOR background\_object IN background\_objects:

SET obj\_rect TO pygame.Rect(background\_object[1][0]-scroll[0]\*background\_object[0],background\_object[1][1]-scroll[1]\*background\_object[0],background\_object[1][2],background\_object[1][3])

IF background\_object[0] EQUALS 0.5:

pygame.draw.rect(display,(14,222,150),obj\_rect)

ELSE:

pygame.draw.rect(display,(9,91,85),obj\_rect)

SET tile\_rects TO []

SET y TO 0

FOR layer IN game\_map:

SET x TO 0

FOR tile IN layer:

IF tile EQUALS '1':

display.blit(dirt\_img,(x\*16-scroll[0],y\*16-scroll[1]))

IF tile EQUALS '2':

display.blit(grass\_img,(x\*16-scroll[0],y\*16-scroll[1]))

IF tile EQUALS '3':

display.blit(flower\_img,(x\*16-scroll[0],y\*16-scroll[1]))

IF tile EQUALS '4':

display.blit(planks\_img,(x\*16-scroll[0],y\*16-scroll[1]))

IF tile EQUALS '5':

display.blit(spruceplanks\_img,(x\*16-scroll[0],y\*16-scroll[1]))

IF tile EQUALS '6':

display.blit(cobble\_img,(x\*16-scroll[0],y\*16-scroll[1]))

IF tile EQUALS '7':

display.blit(leaves\_png,(x\*16-scroll[0],y\*16-scroll[1]))

IF tile EQUALS '8':

display.blit(stone\_img,(x\*16-scroll[0],y\*16-scroll[1]))

IF tile EQUALS '9':

display.blit(wood\_img,(x\*16-scroll[0],y\*16-scroll[1]))

IF tile EQUALS 'p':

display.blit(spruce\_wood\_img,(x\*16-scroll[0],y\*16-scroll[1]))

IF tile != '0':

tile\_rects.append(pygame.Rect(x\*16,y\*16,16,16))

x += 1

y += 1

## Hotbar

pygame.draw.rect(display,(40,40,40),pygame.Rect(50,180,200,20))

IF Currblock EQUALS 'p':

SET Currblockhotbar TO '10'

ELSE:

SET Currblockhotbar TO Currblock

pygame.draw.rect(display,(200,200,0),pygame.Rect((int(Currblockhotbar)\*18)+40,181,20,18))

display.blit(dirt\_img,(60,183))

display.blit(grass\_img,(78,183))

display.blit(flower\_img,(96,183))

display.blit(planks\_img,(114,183))

display.blit(spruceplanks\_img,(132,183))

display.blit(cobble\_img,(150,183))

display.blit(leaves\_png,(168,183))

display.blit(stone\_img,(186,183))

display.blit(wood\_img,(204,183))

display.blit(spruce\_wood\_img,(222,183))

SET player\_movement TO [0,0]

IF moving\_right EQUALS True:

SET player\_movement[0] TO 2

player\_x\_block+=2

IF moving\_left EQUALS True:

player\_movement[0] -= 2

player\_x\_block -=2

player\_movement[1] += vertical\_momentum

vertical\_momentum += 0.2

IF vertical\_momentum > 3:

SET vertical\_momentum TO 3

SET player\_rect,collisions TO move(player\_rect,player\_movement,tile\_rects)

IF collisions['bottom'] EQUALS True:

SET air\_timer TO 0

SET vertical\_momentum TO 0

ELSE:

air\_timer += 1

display.blit(player\_img,(player\_rect.x-scroll[0],player\_rect.y-scroll[1]))

FOR event IN pygame.event.get(): # event loop

IF event.type EQUALS QUIT:

pygame.quit()

sys.exit()

IF event.type EQUALS KEYDOWN:

IF event.key EQUALS K\_RIGHT:

SET moving\_right TO True

IF event.key EQUALS K\_LEFT:

SET moving\_left TO True

IF event.key EQUALS K\_UP:

IF air\_timer < 6:

SET vertical\_momentum TO -5

IF event.key EQUALS K\_1:

SET Currblock TO '1'

IF event.key EQUALS K\_2:

SET Currblock TO '2'

IF event.key EQUALS K\_3:

SET Currblock TO '3'

IF event.key EQUALS K\_4 :

SET Currblock TO '4'

IF event.key EQUALS K\_5:

SET Currblock TO '5'

IF event.key EQUALS K\_6:

SET Currblock TO '6'

IF event.key EQUALS K\_7 :

SET Currblock TO '7'

IF event.key EQUALS K\_8 :

SET Currblock TO '8'

IF event.key EQUALS K\_9:

SET Currblock TO '9'

IF event.key EQUALS K\_0:

SET Currblock TO 'p'

IF event.key EQUALS K\_ESCAPE:

escapeMenu.Main()

IF event.type EQUALS KEYUP:

IF event.key EQUALS K\_RIGHT:

SET moving\_right TO False

IF event.key EQUALS K\_LEFT:

SET moving\_left TO False

## Placing and breaking

SET weird\_glitch\_offset TO -32

scroll\_offset\_x=-100

scroll\_offset\_y=-100

player\_y\_block=player\_rect.y-99

IF pygame.mouse.get\_pressed()[2] EQUALS True:

SET mouse\_pos TO pygame.mouse.get\_pos()

SET mouse\_pos TO np.asarray(mouse\_pos)

mouse\_pos[0]+=player\_x\_block

mouse\_pos[1]+=player\_rect.y-99

SET x\_tile TO int(((mouse\_pos[1]+scroll[1]+weird\_glitch\_offset)/32))

SET y\_tile TO int(((mouse\_pos[0]-32+scroll[0]+player\_y\_block+weird\_glitch\_offset)/32))

IF game\_map[x\_tile][y\_tile]=='0':

SET game\_map[x\_tile][y\_tile] TO Currblock

OUTPUT(Currblock)

IF pygame.mouse.get\_pressed()[0]==True:

SET mouse\_pos TO pygame.mouse.get\_pos()

SET mouse\_pos TO np.asarray(mouse\_pos)

mouse\_pos[0]+=player\_x\_block

mouse\_pos[1]+=player\_rect.y-99

SET x\_tile TO int(((mouse\_pos[1]+scroll[1]+weird\_glitch\_offset)/32))

SET y\_tile TO int(((mouse\_pos[0]-32+scroll[0]+player\_y\_block+weird\_glitch\_offset)/32))

SET game\_map[x\_tile][y\_tile] TO '0'

screen.blit(pygame.transform.scale(display,WINDOW\_SIZE),(0,0))

pygame.display.update()

clock.tick(60)

IF \_\_name\_\_ EQUALS "\_\_main\_\_":

Main()

Menu Program

from tkinter IMPORT W

IMPORT pygame

IMPORT sys

from Button IMPORT Button

IMPORT Platformer

# BACKGROUND

SET WINWIDTH,WINHEIGHT TO 600,400

pygame.init()

SET WIN TO pygame.display.set\_mode((WINWIDTH,WINHEIGHT))

pygame.display.set\_caption('CRINEMAFT')

SET background\_img TO pygame.image.load('Images/backgroundimg.png')

SET PlayButtonImage TO pygame.image.load('Images/PlayButton.png')

SET PlayButton TO Button((WINWIDTH/2)-64,100,PlayButtonImage,128,64)

DEFINE FUNCTION set\_text(string, coordx, coordy, fontSize): #Function to set text

SET font TO pygame.font.Font('freesansbold.ttf', fontSize)

#(0, 0, 0) is black, to make black text

SET text TO font.render(string, True, (255,255,255))

SET textRect TO text.get\_rect()

SET textRect.center TO (coordx, coordy)

RETURN (text, textRect)

DEFINE FUNCTION Draw():

WIN.blit(background\_img, (0,0))

PlayButton.Draw(WIN)

SET totalText TO set\_text("Crine Maft",300, 50, 40)

SET controlsText TO set\_text("Arrow Keys to move,",300, 200, 20)

SET controlsText1 TO set\_text("space to jump," ,300, 250, 20)

SET controlsText2 TO set\_text("right and left click to place and destroy blocks," ,300, 300, 20)

WIN.blit(totalText[0], totalText[1])

WIN.blit(controlsText[0], controlsText[1])

WIN.blit(controlsText1[0], controlsText1[1])

WIN.blit(controlsText2[0], controlsText2[1])

pygame.display.update()

DEFINE FUNCTION Main():

WHILE True:

FOR event IN pygame.event.get():

IF event.type EQUALS pygame.QUIT:

pygame.quit()

sys.exit()

IF event.type EQUALS pygame.KEYDOWN:

IF event.key EQUALS pygame.K\_ESCAPE:

Platformer.Main()

Draw()

IF PlayButton.CheckClick():

Platformer.Main()

IF \_\_name\_\_ EQUALS "\_\_main\_\_":

Main()

Escape Menu

IMPORT pygame

IMPORT sys

from Button IMPORT Button

IMPORT Menu

IMPORT Platformer

SET WINWIDTH,WINHEIGHT TO 600,400

pygame.init()

SET WIN TO pygame.display.set\_mode((WINWIDTH,WINHEIGHT))

SET Back2MenuImg TO pygame.image.load("Images/menu.png")

SET BackGroundImg TO pygame.image.load("Images/backgroundimg.png")

SET ResumeImg TO pygame.image.load("Images/resumebutton.png")

SET ResumeButton TO Button((WINWIDTH/2)-64,100,ResumeImg,128,64)

SET Back2MenuButton TO Button((WINWIDTH/2)-64,200,Back2MenuImg,128,64)

DEFINE FUNCTION set\_text(string, coordx, coordy, fontSize): #Function to set text

SET font TO pygame.font.Font('freesansbold.ttf', fontSize)

#(0, 0, 0) is black, to make black text

SET text TO font.render(string, True, (255,255,255))

SET textRect TO text.get\_rect()

SET textRect.center TO (coordx, coordy)

RETURN (text, textRect)

DEFINE FUNCTION Draw():

WIN.blit(BackGroundImg, (0,0))

Back2MenuButton.Draw(WIN)

ResumeButton.Draw(WIN)

SET totalText TO set\_text("Paused...",300, 50, 40)

WIN.blit(totalText[0], totalText[1])

pygame.display.update()

DEFINE FUNCTION Main():

WHILE True:

FOR event IN pygame.event.get():

IF event.type EQUALS pygame.QUIT:

pygame.quit()

sys.exit()

Draw()

IF Back2MenuButton.CheckClick():

Menu.Main()

IF ResumeButton.CheckClick():

Platformer.Main()

IF \_\_name\_\_ EQUALS "\_\_main\_\_":

Main()

Button

IMPORT pygame

DEFINE CLASS Button:

DEFINE FUNCTION \_\_init\_\_(self, x, y, image, scaleX, scaleY):

SET self.x TO x

SET self.y TO y

SET self.image TO pygame.transform.scale(image, (scaleX, scaleY))

SET self.initialimage TO pygame.transform.scale(image, (scaleX, scaleY))

SET self.rect TO self.image.get\_rect(topleft=(x, y))

SET self.clicked TO False

SET self.Action TO False

SET self.Hover TO False

DEFINE FUNCTION Draw(self, surface):

surface.blit(self.image, (self.rect.x, self.rect.y))

DEFINE FUNCTION CheckClick(self):

SET action TO False

# get mouse position

SET self.pos TO pygame.mouse.get\_pos()

# check mouseover and clicked conditions

IF self.rect.collidepoint(self.pos):

IF pygame.mouse.get\_pressed()[0] EQUALS 1 and self.clicked EQUALS False:

SET self.clicked TO True

SET action TO True

IF pygame.mouse.get\_pressed()[0] EQUALS 0:

SET self.clicked TO False

RETURN action

DEFINE FUNCTION CheckHover(self):

SET self.pos TO pygame.mouse.get\_pos()

IF self.rect.collidepoint(self.pos):

PASS

ELSE:

SET self.Hover TO False

SET self.image TO self.initialimage

IF self.rect.collidepoint(self.pos) and self.Hover EQUALS False:

SET self.image TO self.hoverimage

SET self.Hover TO True

RETURN self.Hover

1. Develop

(13 marks)

* Create a functional application. This stage is where the program code is written. You are to use Python programming language and any other programs suitable for your project. Please attached the code for your application

1. Evaluate (13 marks)

* Develop a survey and test your app with a sample audience
* Base on the response of the sample audience; create an evaluation of your work with graphs and statistics.

Task 2 Software Development Marking Key

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Criteria** | **Possible Mark** | **Your Mark** |
| Programming  Production | **State the Problem**   * Basic description (up to 2 marks) * Detail description (up to 4 marks) * Simple analysis of sample (3 marks) * Detail analysis of samples (up to 6 marks) * Basic description or resources (1 mark) * Details description of resources (up to 3 marks) | 4  6  3 |  |
| **Plan and Design**   * Simple screen layout and sketches (1 mark) * Detail screen layout and sketches (up to 3 marks) * Basic algorithms (5 marks) * Clear and well structure algorithms (up to 10 marks) | 3  10 |  |
| **Develop**   * Basic prototype ( up to 5 marks) * Fully functional prototype ( up to 7 marks) * Fully functional and clear to use (up to 10 marks) * Coding (up to 3 marks) | 10  3 |  |
| **Evaluate**   * Basic personal evaluation (1 mark) * Detail personal evaluation (up to 3 marks) * Basic peer test –survey (up to 2 marks) * Well devlop survey – 10 q’s (up to 5 marks) * Basic peer evaluation (up to 2 marks) * Detail peer evaluation (graphs, percentages, etc) | 3  5  5 |  |
|  | **Total** | 52 |  |

**Teacher comment:**