Univerzitet Donja Gorica

Fakultet primijenjene nauke“ Elektrotehnika i računarstvo” Podgorica

Game Development

Programiranje i algoritmi

Mentor: Ivan Jovović

Ognjen Vojinović 23/041

Valentina Vojinović 23/142

Sadržaj:

Tabela slika:

Abstrakt

Game development je proces stvaranja igrica koji obuhvata dizajniranje, programiranje, umjetnost i testiranje. Timovi obično prolaze kroz nekoliko faza razvoja, uključujući koncipiranje, dizajn igre, razvoj sadržaja, izvršenje i testiranje. Ovo kombinuje tehničke vještine programiranja s kreativnim procesima izrade svijetova i likova. Krajnji cilj je stvoriti zabavno i engaging iskustvo za igrače širom svijeta.

1. Uvod

Videoigrice su postale neizostavan dio savremenog društva, zabavljajući milione ljudi širom svijeta i istovremeno predstavljajući bogatstvo kreativnosti, tehnološke inovacije i kulturnog uticaja. Ubrzan tehnološki napredak posljednjih decenija omogućio je nevjerovatan razvoj industrije videoigara, koja se sada smatra jednom od najdinamičnijih i najprofitabilnijih grana zabavne industrije. Motivacija za odabir teme u game developmentu proizilazi iz više faktora. Aktuelnost teme može biti jedan od ključnih faktora koji istraživača pokreće da odabere određenu temu. Na primjer, moguće je istraživati trendove u industriji igrica, poput popularnosti određenih žanrova ili tehnoloških inovacija poput virtualne stvarnosti (VR) ili proširene stvarnosti (AR).Game development može biti povezana s društvenim, kulturnim ili ekonomskim trendovima, takodje može istraživati kako igrice utiču na mentalno zdravlje ili socijalnu interakciju, ili kako se koriste u obrazovne svrhe. Međutim, uz rastuću popularnost i tehničku složenost, razvoj videoigara takođe je postao sve zahtjevniji proces. Game development obuhvata mnoge discipline, uključujući dizajn, programiranje, umjetnost, muziku i testiranje, a timovi često moraju prolaziti kroz složene izazove kako bi stvorili uspješne igre.

2. Alati i metode

U ovom poglavlju je naveden sav materijal koji je korišćen za implementaciju praktičnog dijela projekta. Materijali su podijeljeni na dva dijela, u prvom dijelu će biti opisan Python, a u drugom PyGame.

2.1. Korišćeni paketi

1. Python je moćan, fleksibilan programski jezik koji se može koristiti za razvoj veb aplikacija, za kreiranje igrica i još mnogo toga. Python je tako osmišljen da ga mogu koristiti i ljudi koji praktično tek ulaze u svijet programiranja.
Vrlo je čitljiv i lagan za upotrebu, a dobra stvar je i to što postoji mnogo objašnjenja odnosno uputstava gdje skoro sve piše o njegovoj primjeni.
Koliko je dobar jezik, govori činjenica da ga koriste neke od najvećih Internet kompanija na svijetu kao što su YouTube, Pinterest, Mozilla, Dropbox, pa i Google u slučaju pretraživanja.

2. Pygame je popularna biblioteka za programiranje igrica u programskom jeziku Python. Sastoji se od kolekcije modula i funkcija koje omogućuju programerima da razvijaju 2D igre na jednostavan i pristupačan način. Ova biblioteka pruža osnovne alate za manipulaciju grafika, zvuka, inputa korisnika i ostalih elemenata potrebnih za izradu videoigara. Pygame se često koristi kao alat za obrazovanje i učenje programiranja. Njegova intuitivna sintaksa i jednostavan pristup omogućuju početnicima da brzo započnu s razvojem svojih prvih igrica, dok naprednije mogućnosti omogućuju iskusnijim programerima da kreiraju kompleksnije projekte.

2.2. Funkcionali i nefunkcionalni zahtjevi sistema

Funkcionalni zahtjevi sistema obično opisuju funkcionalnosti ili operacije koje sistem mora izvršavati. To uključuje sve akcije, procese ili zadatke koje sistem mora podržati kako bi ispunio svoj osnovni cilj. S druge strane, nefunkcionalni zahtjevi opisuju karakteristike ili uslove koji ne direktno definišu funkcionalnosti sistema, već utiču na način na koji sistem izvršava te funkcionalnosti. Ovi zahtjevi često se odnose na performanse, sigurnost, pouzdanost i upotrebljivost sistema.

Primjeri funkcionalnih zahtjeva za game development platformu koje smo koristili za Pygame su:

```
# Strelice za pomjeranje zmije
if event.key == K_UP:
    snake.turn(up)
elif event.key == K_DOWN:
    snake.turn(down)
elif event.key == K_LEFT:
    snake.turn(left)
elif event.key == K_RIGHT:
    snake.turn(right)
```

Slika br 1. Detekcija korisničkog inputa putem tastature ili miša.

```
# Zmija moze da se pomjera u svakom pravcu ako je duzina 1
if len(self.body) == 1:
    self.direction = direction
else:

    # Ako se pomjera lijevo ili desno , osigurati pravac
    # Prva dva "body parta" kvadrata nijesu u istoj y osi
    if direction == left or direction == right:
        if self.body[0][1] != self.body[1][1]:
            self.direction = direction

    # Ako se pomjera gore ili dolje, osigurati isto pravac
    # Prva dva "body parta" kvadrata nijesu u istoj x osi
    if direction == up or direction == down:
        if self.body[0][0] != self.body[1][0]:
            self.direction = direction
```

Slika br 2.Implementaciju logike igre, kao što su pravila igre, mehanike igre i AI.

```
def check_collision(self):

    # Proveriti da li je glava udarila u deo tela
    if self.head in self.body[1:]:
        return True
    else:
        return False
```

```
# Ako sledeca lokacija udari u zid, prebaci se na suprotni zid
next_x = next_x % width
next_y = next_y % height
next_location = (next_x, next_y)
```

Slika br 3.i 4. Mogućnost detekcije sudara između objekata u igri.

3. Rezultati

Tokom izgradnje igrice više smo se bazirali na linearnom modelu (Waterfall) , iz kojeg smo jasno definisali razvoj same igrice. Počevši od samog planiranja i dizajniranja igrice, pa sve do testiranja, implementacije i održavanja. U svakom dijelu faze smo imali svoje specifične ciljeve i aktivnosti koje smo morali dovršiti prije nego što smo prešli na sljedeću fazu. Početni dio se sačinjavao od samih početnih varijabli u kojem smo definisali pojedine osnove kao što su: Ekran, Grid varijable, Boje, Pravac kretanja,FPS. Kasnije sama nadogranja i ukazivanje na greške je dovelo do usavršavanja koda.

```python
import pygame
from pygame.locals import *
import random

pygame.init()

# Ekran igrice
width = 480
height = 480
screen_size = (width, height)
screen = pygame.display.set_mode(screen_size)
pygame.display.set_caption('Snake Game')

# Grid varijable
grid_size = 20
num_rows = width // grid_size
num_cols = height // grid_size

# Pravac kretanja
up = (0, -1)
down = (0, 1)
left = (-1, 0)
right = (1, 0)

# Boje
black = (0, 0, 0)
blue = (25, 103, 181)
dark_green = (67, 160, 71)
light_green = (129, 199, 132)
red = (200, 0, 0)
white = (255, 255, 255)

# Varijable igrice
score = 0
gameover = False

# Sat
clock = pygame.time.Clock()
fps = 10
```

Slika br 5. Početne varijable

3.1. Arhitektura sistema

Arhitektura sistema u kontekstu game developmenta obuhvata organizaciju i strukturu komponenti softvera koje čine igru. Ova arhitektura definiše način na koji su komponente povezane i kako međusobno komuniciraju kako bi podržale funkcionalnosti i zahtjeve igre. Konkretno u našoj igri se nalazi neke od komponenata koje opisuju arhitekturu sistema.

*Game Loop (Igračka petlja):

-Game loop je centralna komponenta arhitekture igre. To je beskonačna petlja koja se izvršava tokom cijelog života igre. U svakoj iteraciji petlje, igra izvršava korake kao što su ažuriranje logike igre, crtanje grafičkih elemenata i obrada korisničkog inputa. Tako i u našoj kao što je prikazano na slici.

```
# Loop
running = True
while running:

    clock.tick(fps)

    # Proveriti akcije dogadjaja
    for event in pygame.event.get():
        if event.type == QUIT:
            running = False
        elif event.type == KEYDOWN:

            # Strelice za pomjeranje zmije
            if event.key == K_UP:
                snake.turn(up)
            elif event.key == K_DOWN:
                snake.turn(down)
            elif event.key == K_LEFT:
                snake.turn(left)
            elif event.key == K_RIGHT:
                snake.turn(right)
```

Slika br 6. Game loop

*Objekti igre:

Objekti igre predstavljaju različite elemente unutra igre, poput likova, neprijatelja, objekata u okolini, efekata itd. Svaki objekt igre može imati svoje ponašanje, svojstva i metode koje se koriste za upravljanje njime, tako i naša zmijica ima metodu upravljanja i ukoliko sama sebe “ujede” završava igra , ima i jos elemenata poput jabuke koja je produžava.

```
def check_collision(self):

    # Proveriti da li je glava udarila u deo tela
    if self.head in self.body[1:]:
        return True
    else:
        return False
```

Slika br 7. Kod za provjeru pokreta i ponasanja

```
# Provera sudara
collision = snake.check_collision()
if collision:
    gameover = True
```

Slika br 8. Kod za provjeru sudara

3.2. Biznis

Game development donosi razne poslovne koristi i prednosti, kako za nezavisne programere tako i za velike razvojne kuće. Jedan od glavnih benefita game developmenta jeste prihod samih prodaja igrica. Uspješne igre mogu generisati značajne prihode od prodaje na različitim platformama poput Steam-a, App Store-a, Google Play-a i konzola. Takođe, kvalitetne igre mogu poslužiti kao snažan marketinški alat za promociju branda ili proizvoda.Kvalitetne i zanimljive igre mogu privući vjerne fanove i zajednice, što može rezultirati dugotrajnim i profitabilnim odnosom s korisnicima. Možda su ostale igrice inovativnog motiva dok naša može poslužiti proširenju platformi i kao kreacija i pokretač zabave mlađem uzrastu.

3.3 Humani aspekti aplikacije

Humani aspekt aplikacije u game developmentu odnosi se na dizajn i implementaciju elemenata igre koji podstiču emocionalnu povezanost, empatiju i dublje razumijevanje između igrača i igre. Tako se može reći da naša igra skreće misli na ne tako dug period ali kao kratotrajna zabava omogućava da u stvarnom životu dolazimo do bržih odluka i refleksa. Kombinacija ovih i drugih elemenata može rezultirati snažnim i dubokim iskustvima za igrače, čineći igre ne samo zabavnim, već i inspirativnim i emocionalno ispunjavajućim iskustvom.

4.Zaključak

U ovom dokumentu na temu „Game Development“ smo objasnili kako se kreira igrica i njen razvoj pomoću Pygame (Python). Praktični dio je bio vezan za implementaciju samog stvaranja videoigre. Pomoćni alat u samom kreiranju videoigre koji nam je olakšao samo razumijevanje i korišćenje uz elementarno poznavanje samog Python-a je bio Pygame koji je sastavni dio Python-a.Samo ispisivanje kodova nije bilo toliko kompleksno već zahtijeva poznavanje gradiva koje smo prešli na univerzitetu. U svakoj varijabli se nalazi najmanji detalj koji pomaže u boljoj mehanizaciji i realizaciji započete ideje, a kasnije ostvarenje cilja.

LITERATURA

[1] "Program Arcade Games With Python And Pygame" autora Paul-a Vincenta Craven-a - Ova knjiga je namijenjena početnicima i nudi korak-po-korak vodič za razvoj igara u Pythonu pomoću Pygame-a. Obuhvata osnove Pygame-a, uključujući osnove grafike, zvuka i interakcije s korisnikom.

[2] "Pygame for Python Game Development" autora Harrison-a Kinsley-a i Will-a McGugan-a - Ova knjiga je priručnik koji obuhvata sve aspekte razvoja igara u Pythonu pomoću Pygame-a. Obuhvata osnove Pygame-a, ali također ide i korak dalje, uključujući složenije koncepte poput umjetne inteligencije u igrama.

[3] „ChatGPT“ AI- vještačka inteligencija koja je pomogla u strukturi teksta.