Raport

1. Skład drużyny: Wiktor Kycia, Jan Topolewski
    a. podział zadań: (co zostało zrobione)
        i. Wiktor: mechanika losowania przeszkód; sterowanie samolotem; mechanizm strzelania; ta sama prędkość obiektów przy różnym fps; grafiki samolotów; aplikacja początkowa odpalająca grę, obsługa pliku settings.txt; grafiki tła i poruszanie się tła; pokazywanie wyników, wyróżnianie aktualnego wyniku, pokazanie wyników tylko z aktualnego poziomu;
        ii. Janek: dźwięki strzelania i muzyka; mechanika pojawiania się skrzynki z ammo; grafiki skrzynek z ammo; mechanika i grafika HP; grafiki i obracanie się kul ognia; mechanika i grafika strzelającego przeciwnika i jego pocisków; easter eggi; mechanizm zatrzymania się gry po straceniu HP i wyświetlanie wyglądu gry po przegranej;
2. Instrukcja i opis gry
    a. nazwa: Niszczyciel asteroid
    b. opis:
    Gra polega na zestrzeliwaniu jak największej liczby lecących w stronę gracza asteroid. Jednocześnie gracz ma ograniczoną ilość amunicji, więc musi zbierać skrzynki z amunicją. Dodatkową uciążliwością na trudnym poziomie jest pociąg Kolei Wielkopolskich, który strzela do gracza. Gracz ma ograniczoną ilość hp, która po zderzeniach z asteroidami i ostrzale od pociągów spada. Po zakończonej grze graczowi gratuluje wyniku nasz Dyrektor. Gracz ma wtedy możliwość rozpoczęcia gry od nowa.
    c. Instrukcja:
    poruszanie: wasd, strzałki, numpad(8, 4, 5, 6)
    strzelanie: spacja, “+” na numpadzie
    odpalenie gry od nowa (po zakończonej grze): x
3. Cele i założenia (pomysły na początku)
    a. gracz jako samolot, który omija przeszkody
    b. przeszkody w postaci elementów latających w powietrzu renderowane w losowych miejscach
    c. gra zawiera poziomy trudności: łatwy, normalny i trudny
    d. gracz może strzelać do obiektów, żeby je zniszczyć
        i. na trudnym poziomie trudności przeszkody (wrogie pociągi) strzelają do gracza

    ii. gracz ma ograniczoną amunicję, która może się w trakcie gry zwiększyć
    iii. odnawianie amunicji (2 metody): powolne odnawianie się, zbieranie paczek z ammo na trasie
  e. Wynik wyświetlany podczas gry, zwiększa się z każdą sekundą i zniszczonym obiektem
  f. najlepsze wyniki zapisywane w pliku razem z nazwą gracza i poziomem trudności (Wiktor)
  g. jeżeli gracz osiągnie wynik, który jest lepszy niż wszystkie inne w historii to wyświetlany jest ekran highscore
  h. gra posiada oryginalną muzykę i dźwięki (strzelania, wybuchów, końca gry itp.)(muzyka - Janek)
  i. grafiki gracza i przeszkód zrobione w gimpie
  j. setup przed grą, w którym gracz może zmieniać rozmaite ustawienia początkowe np.
    i. nazwa użytkownika
    ii. wybór poziomu trudności
    iii. kolor samolotu gracza
  k. setup robiony w bibliotece tkinter, jako oddzielna aplikacja uruchamiająca grę (Wiktor)
4. Problemy i zmiany
  a. Problem 1: (Wiktor) Po wciśnięciu przycisku “Graj” w setup-ie, odpalał się main, ale z błędem, który mówił o tym, że jest brak modułu pygame. Natomiast jak odpalało się main bezpośrednio to wszystko działało.
    Rozwiązanie: Przeinstalowałem pygame, wcześniej miałem pythona w wersji 3.7 (który się trochę różni np. nie ma match case) i tam miałem pygame. Do robienia projektu zainstalowałem python 3.11. Oprócz tego podmieniłem funkcję odpalającą grę z call() na Popen() - to działa, ale uruchamia zminimalizowaną grę
  b. Problem 2: (Wiktor) Przy zapisywaniu wyniku końcowego do pliku wyniki.txt, program zapisywał tę samą linię, aż do momentu zakończenia programu (w każdej klatce wykonał raz)
    Rozwiązanie: Utworzyłem zmienną pomocniczą Ernest, początkowo równą 0, która po zakończeniu gry zwiększa się o 1, przy zapisywaniu do pliku stworzyłem if Ernest == 0 - w ten sposób zapisuje się tylko raz
  c. Problem 3: (Wiktor) czasami występował taki problem, że jak przeszkoda zderzała się z pociskiem lub z graczem, to wywalało całą grę, albo pociski przelatywały przez przeszkodę,

Rozwiązanie: stworzyłem zmienną pomocniczą zniszczona i na jej podstawie program przerywał pętlę iterującą po przeszkodach, przez co nie wyskakuje błąd

5. Wyjaśnienie składni
    a. match case - to samo co switch case w C++
    zastąpienie wielokrotnych elif-ów
    w tym przypadku wypisze “Buzz”

```
x = 1
match x:
    case 0:
        print("Fizz")
    case 1:
        print("Buzz")
```

    b. przechwytywanie wyjątków:
    najpierw program “próbuje” wykonać pewną część kodu i sprawdza, czy nie wystąpi błąd, jeśli tak to wykonuje kod po słowie “except”

```
try:
    print("Hello world!")
except Error():
    print("wystąpił błąd")
```

    c. zwraca wszystkie linie w pliku .txt w formie listy

```
plik.readlines()
```

    d. uruchamia plik python o nazwie main.py

```
from subprocess import *
Popen(["python", "main.py"])
```

    e. tkinter i tkinter.ttk - wbudowane biblioteki python, które pozwalają stworzyć aplikację z UI użytkownika
    Przykładowe kontrolki:
    Label - zwykły tekst
    Frame - ramka
    LabelFrame - ramka z tytułem
    Button - przycisk
    Radiobutton - jednokrotny wybór
    Combobox - jednokrotny wybór z rozwijanej listy
    Scale - wybór liczby z jakiejś tam skali
    Stringvar, Intvar - klasy, które imitują zmienne, można je przypisać do powyższych elementów, posiadają atrybuty get i set
    więcej info (nie chce mi sie pisać):
    https://www.obliczeniowo.com.pl/496

f. Służy do odtwarzania muzyki w tle i dźwięków strzelanin oraz wybuchów

```
background_music = "interweb.mp3"
mixer.init()
wybuch_sound = mixer.Sound('wybuch.WAV')
wybuch_sound.set_volume(0.1)
shot_sound = mixer.Sound(shot_sound_path)
shot_sound.set_volume(0.2)
mixer.music.load(background_music)
mixer.music.set_volume(0.4)
mixer.music.play(-1)
```

g. Służy do animacji obracania asteroidy i jej wyświetlania

```
def draw(self):
        self.ksztalt = pygame.Rect(self.x, self.y, self.width,
self.height)
        obrocony_image = pygame.transform.rotate(self.image,
self.angle)
        self.angle += 20 * dt
        if self.asteroida == 1:
            nowy_rect = obrocony_image.get_rect(center =
self.image.get_rect(center = (self.x + 25, self.y + 25)).center)
            screen.blit(fire_small, (self.x, self.y))
        if self.asteroida == 2:
            nowy_rect = obrocony_image.get_rect(center =
self.image.get_rect(center = (self.x + 40, self.y + 40)).center)
            screen.blit(fire_huge, (self.x, self.y))
        screen.blit(obrocony_image, nowy_rect)
```

6. Podsumowanie

“No fajnie było, przyjemnie robiło się grę”