**Spis treści**

1. Tytuł aplikacji
2. Rodzaj aplikacji
3. Opis aplikacji
4. Klasy
   1. Dane
      1. Zmienne
         1. ListaStartowa
         2. ListaMiejsc
         3. ListaFinalowa
         4. PozycjaPionkow
         5. OryginalnePolaPionkow
         6. Okrazenia
         7. AktualnyPionek
      2. Metody
         1. Konstruktor
   2. Wizualizacja
      1. Zmienne
         1. PolaGry
         2. plansza
         3. ListaPionkow
         4. Pomocnicza
         5. Pionek
      2. Metody
         1. Konstruktor
   3. MenadzerGry
      1. Zmienne
         1. PionkiGry
         2. PolaGry
         3. losowePrzesuniecie
         4. AktualnyGracz
         5. LosTekts
         6. PrzyciskStart
         7. PrzyciskWyjsc
         8. PrzyciskLosowania
      2. Metody
         1. Konstruktor
         2. RozpocznijGre
         3. Zamknij
         4. Gra
         5. PokazaniePrzyciskuLosuj
         6. Losowanie
         7. RozpocznijPrzesuwaniePionka
         8. UstawPozycjePionka
         9. UstwaPozycjeFinalowa
         10. OknoFinalowe
5. Grafika
   1. Modele
      1. CzerwonyPionek
      2. ZoltyPionek
      3. ZielonyPionek
      4. NiebieskiPionek
      5. Plansza
6. Charakterystyka aplikacji
7. **Tytuł aplikacji**

Chinczyk

1. **Rodzaj Aplikacji**

Gra 3D; Gracz kontra Gracz; Lokalna rozgrywka; Gra imprezowa

1. **Opis Aplikacji**

Gra 3D wzorowana na zasadach gry planszowej Chińczyk. W grze uczestniczy 4 graczy, gdzie każdy z nich steruje odpowiednimi pionkami. Celem gracza jest dostarczenie wszystkich jego pionków na centrum planszy. By tego dokonać, trzeba okrążyć całą planszą pionkiem, by potem wejść na miejsca finałowe. Gracz losuje jak wiele pól może pokonać pionek w swojej turze, dzięki przyciskowi losowania. Gra się zakończy gdy jeden z gracz dostarczy wszystkie swoje pionki na odpowiednie (dla jego koloru) miejsca finałowe.

1. **Klasy**
   1. **Dane**

Ta klasa posiada wszystkie dane potrzebne dla MenadzeraGry. Większość z zmiennych jest w postaci list.

* + 1. **Zmienne**

ListaStartowa - Pozycja początkowa pionków. Zmienna składa się z 4 list, a każda z nich posiada 4 zmienne trzy liczbowe. Każda z list jest przypisana do danego gracza. I - Czerwony, II - Zółty, III - Zielony, IV - Niebieski. Zaś każda taka lista posiada 4 zmienne dla każdego z pionków. pierwsza liczba z zmiennej to oś X, druga, zaś Y, trzecia to Z.

self.ListaStartowa = [

[

(-20.5, -26, 3), (-20.5, -30, 3), (-25.5, -30, 3), (-25.5, -26, 3)

],

[

(-24, 18.5, 3), (-28.5, 18.5, 3), (-24, 24, 3), (-28.5, 24, 3)

],

[

(22, 21, 3), (22, 25, 3), (27, 25, 3), (27, 21, 3)

],

[

(22.5, -23.5, 3), (27, -23.5, 3), (22.5, -28, 3), (27, -28, 3)

]

]

ListaMiejsc - Lista wszystkich pozycji na planszy. Są to zmienne trzy liczbowe. Pierwsza z nich to liczba wskazująca oś X, druga zaś Y, trzecia to Z.

self.ListaMiejsc = [

(-5.5, -22, 3), (-5.5, -16.5, 3), (-5.5, -11.5, 3), (-5.5, -6, 3), (-11, -6, 3), (-16, -6, 3),

(-21.5, -6, 3), (-26.5, -6, 3), (-26.5, -1, 3), (-26.5, 5, 3), (-21.5, 5, 3), (-16, 5, 3), (-11, 5, 3),

(-5.5, 5, 3), (-5.5, 10, 3), (-5.5, 15, 3), (-5.5, 21, 3), (-5.5, 26, 3), (0, 26, 3), (5.5, 26, 3),

(5.5, 21, 3), (5.5, 16, 3), (5.5, 10.5, 3), (5.5, 5, 3), (11, 5, 3), (16, 5, 3), (21.5, 5, 3), (26.5, 5, 3),

(26.5, -0.5, 3), (26.5, -6, 3), (21.5, -6, 3), (16, -6, 3), (11, -6, 3), (5.5, -6, 3), (5.5, -11.5, 3),

(5.5, -16.5, 3), (5.5, -22, 3), (5.5, -27.5, 3), (0, -27.5, 3), (-5.5, -27.5, 3)

]

ListaFinalowa - Lista pozycji końcowych pionków. Zmienna składa się z 4 list, zaś te listy posiadają 4 zmienne trzy liczbowe. Każda lista odpowiada odpowiedniemu kolorowi gracza I - Czerwony, II - Zółty, III - Zielony, IV - Niebieski.. Zmienne liczbowe zaś to miejsca finałowe dla pionków danego koloru. Każda z liczb zmiennych trzycyfrowych wskazuje na układ współrzędnych. Pierwsza liczba wskazuje oś X, druga zaś Y, trzecia to Z.

self.ListaFinalowa = [

[

(0, -22, 3), (0, -16.5, 3), (0, -11.5, 3), (0, -6, 3)

],

[

(-21.5, -1, 3), (-16, -1, 3), (-11, -1, 3), (-5.5, -1, 3)

],

[

(0, 21, 3), (0, 16, 3), (0, 10, 3), (0, 5, 3)

],

[

(21.5, -1, 3), (16, -1, 3), (10.5, -1, 3), (6, -1, 3)

]

]

PozycjaPionkow - Lista składająca się z 4 zmiennych, których każda z nich to liczba. Podstawowo zmienne ustawione są na liczby początkowych pozycji. Wraz z postępem w grze zmienne ulegają zwiększeniu, aż do momentu do którego zmienna będzie równa indeksowi ostatniej pozycji z ListaMiejsc. Wtedy zmienne są resetowane i proces zaczyna się od początku. Każda zmienna obrazuje danego gracza. Pierwsza to czerwony, druga to żółty, trzecia to zielony, zaś czwarta to niebieski.

OryginalnePolaPionkow - Lista składająca się z 4 zmiennych, których każda z nich to liczba. Są tam przetrzymywane indeksy startowe dla ListyMiejsc dla nowych pionków. Każda zmienna obrazuje danego gracza. Pierwsza to czerwony, druga to żółty, trzecia to zielony, zaś czwarta to niebieski.

Okrazenia - Lista składająca się z 4 zmiennych, których każda z nich to liczba. Obrazuje to kiedy pionek będzie szedł na miejsca z ListaFinałowa, a kiedy jeszcze musi zrobić okrążenie. 1 - Robi okrążenie planszy; 2 - Idzie na miejsca finałowe. Każda zmienna obrazuje danego gracza. Pierwsza to czerwony, druga to żółty, trzecia to zielony, zaś czwarta to niebieski.

AktualnyPionek - Lista składająca się z 4 zmiennych, których każda z nich to liczba. Obrazują one który aktualnie pionek jest używany przez danego gracza. Każda zmienna obrazuje danego gracza. Pierwsza to czerwony, druga to żółty, trzecia to zielony, zaś czwarta to niebieski.

* + 1. **Metody**

Konstruktor - Metoda ma za zadanie wygenerować zmienne, które są używane podczas gry.

* 1. **Wizualizacja**

Klasa odpowiedzialna za graficzną stronę gry. Są tam tworzone pionki gracz, jak również przechowywane, plansza gry, jak również określone ustawienia kamery. Dziedziczy ona od klasy ShowBase.

* + 1. **Zmienne**

PolaGry - Obiekt klasy Dane

plansza - Obiekt graficzny, który posiada właściwości modelu Plansza.

ListaPionkow - Lista składająca się z 4 list, w każdej z nich są 4 zmienne “x”. Obrazują one obiekty pionków graczy. Każda z list jest przypisana do danego gracza. I - Czerwony, II - Zółty, III - Zielony, IV - Niebieski

self.ListaPionkow = [

["x", "x", "x", "x"],

["x", "x", "x", "x"],

["x", "x", "x", "x"],

["x", "x", "x", "x"]

]

Pomocnicza - Zmienna używana do operowania na ListaPionkow. Obrazuje ona który pionek z listy jest teraz używany.

Pionek - Obiekt graficzny, który posiada właściwości modelu CzerwonyPionek, ZoltyPionek, ZielonyPionek i NiebieskiPionek. Zmienna odpowiednio zmienia model zależnie od listy na której operuje.

* + 1. **Metody**

Konstruktor - Wykonuje wszystkie operacje graficzne. Od ustawienia kamery do wygenerowania obiektów graficznych i ustawienia ich w odpowiednie miejsce. Na samym początku metoda uruchamia konstruktor klasy ShowBase. Następnie tworzony jest obiekt PolaGry klasy Dane.

super().\_\_init\_\_()

self.PolaGry = Dane()

Potem ustawiona jest kamera w odpowiedniej pozycji i wyłączana jest możliwość poruszania ją.

self.disable\_mouse()

self.camera.setPos(0, -100, 100)

self.camera.setHpr(0, -45, 0)

Potem zaś generowany jest model planszy.

plansza = self.loader.loadModel("Modele/Plansza.bam")

Następnie następuje procedura wygenerowania modeli Pionek.

for i in range(16):

if i%4 == 0:

Pomocnicza += 1

try:

Pionek = self.loader.loadModel("Modele/CzerwonyPionek.bam")

self.ListaPionkow[0][Pomocnicza] = Pionek

except OSError as e:

print(e)

break

except UnboundLocalError as ULE:

print(ULE)

break

elif i%4 == 1:

try:

Pionek = self.loader.loadModel("Modele/ZoltyPionek.bam")

self.ListaPionkow[1][Pomocnicza] = Pionek

except OSError as e:

print(e)

break

except UnboundLocalError as ULE:

print(ULE)

break

elif i%4 == 2:

try:

Pionek = self.loader.loadModel("Modele/ZielonyPionek.bam")

self.ListaPionkow[2][Pomocnicza] = Pionek

except OSError as e:

print(e)

break

except UnboundLocalError as ULE:

print(ULE)

break

else:

try:

Pionek = self.loader.loadModel("Modele/NiebieskiPionek.bam")

self.ListaPionkow[3][Pomocnicza] = Pionek

except OSError as e:

print(e)

break

except UnboundLocalError as ULE:

print(ULE)

break

Na końcu zaś przypisywana jest lokalizacja każdego z pionków i następuje ich wyrenderowanie. Renderowana jest również plansza.

plansza.reparentTo(self.render)

for i in range(16):

if i%4 == 0:

Pomocnicza += 1

self.ListaPionkow[0][Pomocnicza].reparentTo(self.render)

self.ListaPionkow[0][Pomocnicza].setPos(self.PolaGry.ListaStartowa[0][Pomocnicza])

elif i%4 == 1:

self.ListaPionkow[1][Pomocnicza].reparentTo(self.render)

self.ListaPionkow[1][Pomocnicza].setPos(self.PolaGry.ListaStartowa[1][Pomocnicza])

elif i%4 == 2:

self.ListaPionkow[2][Pomocnicza].reparentTo(self.render)

self.ListaPionkow[2][Pomocnicza].setPos(self.PolaGry.ListaStartowa[2][Pomocnicza])

else:

self.ListaPionkow[3][Pomocnicza].reparentTo(self.render)

self.ListaPionkow[3][Pomocnicza].setPos(self.PolaGry.ListaStartowa[3][Pomocnicza])

* 1. **MenadzerGry**

Główna klasa gry. Posiada wszelkie metody, oraz elementy UI (User Interface) gracza. Klasa dziedziczy po klasie Onscreentext.

* + 1. **Zmienne**

PionkiGry - Obiekt klasy Wizualia

PolaGry - Obiekt klasy Dane

losowePrzesuniecie - Zmienna posiadająca wartość wylosowaną przez gracza.

AktualnyGracz - zmienna typu integer, obrazuje jaki gracz aktualnie gra.

TekstObiekt - Obiekt UI, posiada tekst Chińczyk pozycję równą 0 = x i 0.5 = z, oraz skalę 0.4.

self.TekstObiekt = OnscreenText(text="Chińczyk", pos=(0, 0.5), scale=0.4)

LosTekst - Obiekt UI, posiada tekst “Tu będzie liczba losowana” i skalę 0.2  
self.LosTeskt = OnscreenText(text="Tu będzie liczba losowana", scale=0.2)

PrzyciskStart - Obiekt UI, posiada tekst “Graj” pozycję równą 0 = x, 0 = y , 0.1 = z. Skalę równą 0.08. Po naciśnięciu obiektu wywoływana jest funkcja RozpocznijGre

self.PrzyciskStart = DirectButton(text=("Graj"), scale=0.08, command=self.RozpocznijGre, pos=(0, 0, 0.1))

PrzyciskWyjsc - Obiekt UI, posiada tekst “Wyjdź” pozycję równą 0 = x, 0 = y , -0.1 = z. Skalę równą 0.08. Po naciśnięciu obiektu wywoływana jest funkcja Zamknij

self.PrzyciskWyjsc = DirectButton(text=("Wyjdź"), scale=0.08, command=self.Zamknij, pos=(0, 0, -0.1))

PrzyciskLosowania - Obiekt UI, posiada tekst “Losuj” Skalę równą 0.01. Po naciśnięciu obiektu wywoływana jest funkcja Losowanie

self.PrzyciskLosowania = DirectButton(text=("Losuj"), scale=0.1, command=self.Losowanie)

Ostatni - Zmienna typu boolean. Określa czy pionek, który aktualnie gracz steruje osiągnął ostatnie wolne miejsce z ListaFinalowa.

Pozycja - Zmienna trój liczbowa oznaczająca pozycję pionka w układzie współrzędnych. Zmienna przyjmuje wartość z listy ListaMiejsc o indeksie PozycjaPionkow od AktualnyGracz. Pierwsza liczba w ciągu oznacza oś X, druga oznacza oś Y, zaś trzecia oś Z.

* + 1. **Metody**

Konstruktor - Generuje wszystkie zmienne potrzebne do działania gry. Uruchamia również konstruktor klasy OnscreenText

super().\_\_init\_\_()

oraz uruchamia PionkiGry jako graficzną reprezentację gry.

self.PionkiGry.run()

Ukrywane są również elementy UI takie jak LosTekst i PrzyciskLosowania

self.LosTeskt.hide()

self.PrzyciskLosowania.hide()

RozpocznijGre - Po wywołaniu metoda niszczy obiekt PrzyciskStart i chowa PrzyciskWyjsc i TekstObiekt. Następnie wywołuje funkcję Gra.

def RozpocznijGre(self):

self.PrzyciskStart.destroy()

self.PrzyciskWyjsc.hide()

self.TekstObiekt.hide()

self.Gra()

Zamknij - Metoda, która zamyka program.

def Zamknij(self):

sys.exit()

Gra - Metoda ma za zadanie ustawić pierwsze pionki wszystkich graczy na ich pola startowe.

for i in range(4):

self.PionkiGry.ListaPionkow[i][0].setPos(self.PolaGry.ListaMiejsc[self.PolaGry.OryginalnePolaPionkow[i]])

Następnie po sekundzie uruchamia funkcję PokazaniePrzyciskuLosuj z komunikatem losuj, dzięki metodzie taskMgr (Task Manager) doMethodLater.

taskMgr.doMethodLater(1, self.PokazaniePrzyciskuLosuj, name = "Losuj")

PokazaniePrzyciskuLosuj - Metoda odsłania obiekt PrzyciskLosowania i modyfikuje wartość zmiennej AktualnyGracz. Najpierw zostaje do niej dodana wartość 1 potem zaś wyciągany jest modulo 4 z nowo utworzonej liczby.

self.PrzyciskLosowania.show()

self.AktualnyGracz = (self.AktualnyGracz+1)%4

Losowanie - Metoda służy do wylosowania liczby obrazującej jak wiele pól pionek gracza przejdzie. Najpierw ukrywany jest PrzyciskuLosowania

self.PrzyciskLosowania.hide()

Potem zaś następuje losowanie liczby. Dla realizmu liczba losowana ma zakres od 1 do 6, jak w kościach do gry

self.LosowePrzesuniecie = random.randint(1, 6)

Następnie zamieniany jest tekst obiektu LosTekst i jest on odkrywany.

self.LosTeskt.text = 'Wylosowano\n' + str(self.LosowePrzesuniecie)

self.LosTeskt.show()

Na koniec po 2 sek uruchamiana jest funkcja RozpocznijPrzesuwaniePionka z komunikatem ”Rozpocznij przesuwanie pionka”. Tutaj znowu jest używany obiekt taskMgr z metodą doMethodLater

taskMgr.doMethodLater(2, self.RozpocznijPrzesuwaniePionka, name='Rozpocznij przesuwanie pionka')

RozpocznijPrzesuwaniePionka - Metoda ta jest główną metodą przesunięcia pionka. Jest to najbardziej rozbudowana metoda w całym programie. Metoda zawiera bloki warunkowe, pętla jak i metody z biblioteki Task.

Metoda rozpoczyna się od schowania LosTekst.

self.LosTeskt.hide()

Następnie tworzona jest pętla od zmiennej LosowePrzesuniecie. Na początku pętli zmienna Ostatni ustawiana jest na False i sprawdzane jest czy aktualna pozycja pionka jest taka sama jak pozycja początkowa pozycja pionka. Jeśli tak to oznacza, że pionek rozpoczyna kolejne okrążenie planszy.

for i in range(self.LosowePrzesuniecie):

self.Ostatni = False

if self.PolaGry.ListaMiejsc[self.PolaGry.PozycjaPionkow[self.AktualnyGracz]] == self.PolaGry.ListaMiejsc[self.PolaGry.OryginalnePolaPionkow[self.AktualnyGracz]]:

self.PolaGry.Okrazenia[self.AktualnyGracz] += 1

Następnie jeśli zmienna na liście Okrazenia od indeks AktualnyGracz nie jest równa dwa. Jeśli tak to oznacza, że jest to pierwsze okrążenie pionka i powinien on przejść całą planszę.

if self.PolaGry.Okrazenia[self.AktualnyGracz] != 2:

Później sprawdzana jest pozycja pionka. Jeśli ta pozycja zwiększona o 1 będzie równa długości ListaMiejsc. To oznacza, że pionek doszedł do końca listy i trzeba zresetować jego pozycję do zera. Jeśli jednak jest odwrotnie to wartość PozycjaPionka od indeksu AktualnyGracz jest zwiększana o 1.

if self.PolaGry.PozycjaPionkow[self.AktualnyGracz] + 1 == len(self.PolaGry.ListaMiejsc):

self.PolaGry.PozycjaPionkow[self.AktualnyGracz] = 0

else:

self.PolaGry.PozycjaPionkow[self.AktualnyGracz] += 1

Po ustaleniu pozycji pionka. Tworzona jest zmienna Pozycja.

Pozycja = self.PolaGry.ListaMiejsc[self.PolaGry.PozycjaPionkow[self.AktualnyGracz]]

Następnie tworzone jest zadanie z wywołaniem funkcji UstawPozycjePionka o komunikacie “Przesuniecie Pionka”. Ustawiamy opóźnienie zadania o wartości i (zmienna iteracyjna z pętli). Na końcu wywoływana jest metoda taskMgr (Task Manager) add. Gdzie za Task podajemy zadanie oraz extraArgs (extract Argument) podajemy zmienną Pozycja.

Zadanie = Task(self.UstawPozycjePionka, name='Przesuniecie Pionka')

Zadanie.setDelay(i)

taskMgr.add(Zadanie, extraArgs=[Pozycja])

I pętla rozpoczyna się od nowa.

Jeśli jednak pionek ukończył swe okrążenie i teraz powinien iść na miejsca finałowe to wtedy, dzieje się druga część metody.

Sprawdzana jest zmienna z listy PozycjaPionkow o indeksie AktualnyGracz. Jeśli jest ona większa od 4 wtedy ta zmienna jest ustawiona na -1. Następnie sprawdzany jest warunek czy pozycja pionka nie jest równa ostatniemu indeksowi z listy finałowej dla aktualnego gracza. Jeśli tak zmienna Ostatni ustawiana jest na True. Jeśli nie to pozycja pionka jest zwiększana o 1.

if self.PolaGry.PozycjaPionkow[self.AktualnyGracz] > 4:

self.PolaGry.PozycjaPionkow[self.AktualnyGracz] = -1

if self.PolaGry.PozycjaPionkow[self.AktualnyGracz] == len(self.PolaGry.ListaFinalowa[self.AktualnyGracz][self.PolaGry.AktualnyPionek[self.AktualnyGracz]]) - 1:

self.Ostatni = True

else:

self.PolaGry.PozycjaPionkow[self.AktualnyGracz] += 1

Następnie powtarzany jest proces tworzenia pozycji pionka, jednak brana pod uwagę jest ListaFinalowa zamiast ListaMiejsc.

Pozycja = self.PolaGry.ListaFinalowa[self.AktualnyGracz][self.PolaGry.PozycjaPionkow[self.AktualnyGracz]]

Zadanie = Task(self.UstwaPozycjeFinalowa, name = "Przesuniecie finalowe")

Zadanie.setDelay(i)

taskMgr.add(Zadanie, extraArgs=[Pozycja])

Na samym końcu metoda sprawdza, czy długość ListaFinalowa o indeksie AktualnyGracz jest równa 0. Jeśli tak to znaczy, że któryś z graczy wygrał i wywoływana jest Metoda OknoFinalowe z opóźnieniem i+1 i komunikatem “Koniec Gry”. Jeśli jednak nie to wywoływana jest metoda PokazaniePrzyciskuLosuj z opóźnieniem i+1 i komunikatem “Tura od początku”.

if len(self.PolaGry.ListaFinalowa[self.AktualnyGracz]) == 0:

taskMgr.doMethodLater(i+1, self.OknoFinalowe, name = "Koniec Gry")

else:

taskMgr.doMethodLater(i+1, self.PokazaniePrzyciskuLosuj, name = "Tura od poczatku")

UstawPozycjePionka(Pozycja) - Metoda ma za zadanie ustawić odpowiedni pionek na planszy gry. Metoda przyjmuje parametr Pozycja, który określa punkt na którym pionek ma być umiejscowiony.

def UstawPozycjePionka(self, Pozycja):

self.PionkiGry.ListaPionkow[self.AktualnyGracz][self.PolaGry.AktualnyPionek[self.AktualnyGracz]].setPos(Pozycja)

UstawPozycjeFinalową(Pozycja) - Metoda ma za zadanie ustawić odpowiedn pionek na miejscach finałowych danego gracza. Metoda przyjmuje parametr Pozycja, który określa punkt na którym pionek ma być umiejscowiony.

def UstwaPozycjeFinalowa(self, Pozycja):

self.PionkiGry.ListaPionkow[self.AktualnyGracz][self.PolaGry.AktualnyPionek[self.AktualnyGracz]].setPos(Pozycja)

Po ustawieniu pionka metoda sprawdza, czy zmienna Ostatni jest równa True.

if self.Ostatni == True:

Jeśli tak oznacza to, że następne ruchy gracza będą oddziaływały na nowy pionek. W tym celu do listy AktualnyPionek o indeksie od AktualnyGracz dodawana jest wartość 1, zmienna zapisana na liście Okrazenia od indeksu AktualnyGracz jest resetowana do zera i pole, na które doszedł pionek jest usuwane z listy ListaFinalowa.

del self.PolaGry.ListaFinalowa[self.AktualnyGracz][-1]

self.PolaGry.AktualnyPionek[self.AktualnyGracz] += 1

self.PolaGry.Okrazenia[self.AktualnyGracz] = 0

Również podstawiany jest nowy pionek na pole startowe na planszy i zmienna zapisana na liście PozycjaPionkow od AktualnyGracz jest resetowana do stanu sprzed ruchu pionkiem.

self.PionkiGry.ListaPionkow[self.AktualnyGracz][self.PolaGry.AktualnyPionek[self.AktualnyGracz]].setPos(self.PolaGry.ListaMiejsc[self.PolaGry.OryginalnePolaPionkow[self.AktualnyGracz]])

self.PolaGry.PozycjaPionkow[self.AktualnyGracz] = self.PolaGry.OryginalnePolaPionkow[self.AktualnyGracz]

OknoFinalowe - Metoda sprawdza, który gracz doszedł wszystkimi swoimi pionkami na miejsca finałowe. Zależnie od wartości AktualnyGracz zmienia tekst obiektu LosTekst.

if self.AktualnyGracz == 0:

self.LosTeskt.text = "Wygrana\n Czerwonego"

elif self.AktualnyGracz == 1:

self.LosTeskt.text = "Wygrana\n Zoltego"

elif self.AktualnyGracz == 2:

self.LosTeskt.text = "Wygrana\n Zielonego"

else:

self.LosTeskt.text = "Wygrana\n Niebieskiego"

Następnie odkrywa obiekt LosTekt i odkrywa obiekt PrzyciskWyjsc

self.LosTeskt.show()

self.PrzyciskWyjsc.show()

1. **Grafika**

Wszelkie modele zostały stworzone dzięki oprogramowaniu blender. Przechowywane są one w folderze Modele i wykorzystywane są przez klasę wizualia. Jednak Panda 3D nie wspiera format pliku .blend ani .fbx. By uporać się z tym problemem wykorzystany został program blend 2 bam, który konwertuje plik do formatu bam, który jest wspierany przez Panda 3D.

* 1. **Modele**

|  |  |  |
| --- | --- | --- |
| **Nazwa** | **Model** | **Tekstura** |
| CzerwonyPionek |  |  |
| ZoltyPionek |  |  |
| ZielonyPionek |  |  |
| NiebieskiPionek |  |  |
| Plansza |  |  |

1. **Charakterystyka aplikacji**

Język programowania: Python 3.8.

Wykorzystane biblioteki: random, sys, Panda 3D: (OnscreenText, ShowBase, loadPrcFile, Task).

Narzędzia użyte do stworzenia aplikacji: blender, gimp, Pycharm Community Edition 2020.3.2, Dokumenty google.

Narzędzia potrzebne do uruchomienia aplikacji: Jakikolwiek kompilator pythona z zainstalowaną biblioteką Panda 3D.