**Uniwersytet Jagielloński w Krakowie**

Wydział Fizyki, Astronomii i Informatyki Stosowanej

**Paweł Kołodziej**

Nr albumu: 1153732

**Tytuł pracy dyplomowej**

Praca magisterska  
na kierunku Informatyka Gier Komputerowych

Praca wykonana pod kierunkiem

Dr Anny Sochockiej

<Instytut/Zakład>

Kraków 2023

**Spis treści**

1. **Wstęp                                                                                                                              x**
   1. Kontekst……………………………………………………………………………x
   2. Zakres pracy…………………………………………………………...x
2. **Struktura pracy                                                                                                             x**
   1. Cel pracy…………………………………………………………………………...x
3. **Budowa mózgu                                                                                                              x**

3.1 Budowa mózgu…………………………………………………………………….x

1. **Gry poważne                                                                                                                  x**
   1. Gry poważne……………………………………………………………………….x
2. **Opis implementacji                                                                                                       x**
   1. Użyte narzędzia…………………………………………………………………….x
   2. Implementacja……………………………………………………………………...x
   3. Tutorial dla użytkownika…………………………………………………………..x
3. **Opis testów i badań                                                                                                       x**
4. **Podsumowanie                                                                                                               x**
5. **Bibliografia                                                                                                                    x**

Rozdział 1

**Wstęp**

1. **Kontekst**

Ludzki mózg jest bardzo skomplikowanym elementem naszego ciała. Istnieje wiele dziedzin nauki poświęconych zgłębianiu zasad na których oparte jest funkcjonowanie tego narządu. Nie jest to jednak tematyka zarezerwowana tylko dla osób specjalizujących się w anatomii czy psychologii. Poznawanie fundamentów na których opiera się działanie ludzkiego mózgu oraz układu nerwowego jest nie tylko bardzo ciekawe ale może być bardzo ważne dla każdego z nas w kontekście zrozumienia mechanizmów które nami kierują. Możliwe jest dzięki temu łatwiejsze zrozumienie dlaczego my jako ludzie różnimy się od siebie, czym mogą być spowodowane problemy w funkcjonowaniu oraz jak ważne jest zbadanie o własny organizm i dostarczanie mu cennych składników, które są tak ważne dla zdrowego funkcjonowania naszego organizmu . Bardzo ważnym aspektem jest sprawienie aby nauka była jak najbardziej przystępna i atrakcyjna dla młodych osób.  Moze mloda osoba w ten sposób może nie tylko zrozumieć jak funkcjonuje jej ciało, ale może być to bodziec do głębszego zainteresowania się takimi dziedzinami nauki jak psycholgia, medycyjna, kognitywityka czy nauki związane z technologią.

Cześć implementacyjna niniejszej pracy magisterskiej została wykonanna w formie gry poważnej, i ma za zadanie wskazać jak gry poważne, oraz niektóre jej jej elementy sa w stanie ułatwić proces uczenai si ei poznawania nowych zagadnień. Gra została wykonanan w silniku Unity, atomiast modele mózgu zostały wykonane w programie do grafiki trójwymiarowej Blender.      Praca zostałą podzielona na rozdiały, które przybliżają strukturę pracy magisterskiej, szczegóły implementacji, uzasadnią wybór narzędzi oraz tematyki. Przybliżona zostanie również kwestia samych gier poważnych, zjawiska grywalizacji, teoretycznej strony budowy mózgu. Praca zwieńczona jest testem i badaniem  na użytkownikachm dotyczącym efektywnośći uczenia issię za pomocą takiej gry. Na końcu znajduje się tez bibliografia.

**Wstęp**

1. **Zakres pracy**

Celem pracy było wykonanie gry poważnej, któ®a umożliwia użytkownikowi zapozanie się z działąniemm i budowa ludzkiego mózgu. Zaprezentwoanie mózgu w formie interaktywnego trówjymairowego modelu miało za zadanie lepiej zobrazować użytkownikowi strukturę i budowę mózgu, a quiz i test wiedzy połączony z rankingiem wsyztskich użytkowników miał na calu wprowadzenie podstawowych mechanizmó grwazlizacji do gry. Zakres gry prezentuje się w następujący sposób:

- w menu głównym, gracz ma do wyboru zapoznanie siez budową i funcjonowaniem ludzkiego mózgu, lub przejść do bardziej wewntrznej części i zapoznać się z budową neuronu i fundamentalnych mechanizmóstojacyh za funkcjonowaniem mózgu,

- gracz w kolejnych ekranach ma możliwość interkacji z trójwymiarowym modelem mózgu, ma możliwość obracania modelu, zwiększania zmniejdszania, a takżę wyboru poszczególnych elementów, i wyświetelenia odpowiedniej informacji dotycznącej wybranego fragmentu mózgu,

- gracz ma możliwość przystąpienia do testu wiedzy, któ®y ma na celu sprawdzenie i przetestowanie czy udało się użytkownikowi przybliżyć pewne zagadnienia doyczace mżógu,

- gracz ma możliwość podglądau rankingu wynikó innych graczy,

1. **Wstęp + omówienie tego czego dotyczy praca (część pisemna) + struktura części pracy**

d

1. **Wstęp + omówienie tego co jest w części praktycznej**

d

1. **Mózg teoria**

Mózg pełni niezbędną rolę w unkcjonowaniu każdego człowikeka. Składa się z rócnych struktur, któ®e są zroganizowane i połączone w całość, a między nimi występuje odpoweidni hierarchia. Opis anatomii ludzkiego móżg można zacząć od kory mózgowej – jest to najbardziej zewnetrzna warstwa mózgu, któ®a okrywa swoją powierzchnią półkule. Skłąda się z wielu neuronów i stanowi cześć istoty szarej układu erwowego. dwóch półku – lewej i prawej. Półkule są ze sobą połączone spoidłę wielki,m. Kazdappółkula pełni swoje charakterystyczne funkcje. Lewa poółkula odpowiada bardziej za logiczne myslenie,… prawa za kreatywne…. Następnie każda z półkul słąda się z różnych płatów: czołowego, skroniowego, ciemieniowego, potylicznego. Płąty są następnie podzielone na mniejsze obszary, wśród których każdy z nich odpowiada za inne potrzebne do funkcjonowania role.

Neuron, podstawowa jednostka funkcjonalna naszego układu nerwowego, odgrywa kluczową rolę w przekazywaniu sygnałów elektrycznych w naszym ciele. Zrozumienie, jak działa neuron, pompa sodowo-potasowa i przewodnictwo elektryczne jest niezwykle istotne dla naszego poznania mózgu i jego funkcjonowania.

Neuron jest zbudowany z trzech głównych części: dendrytów, ciała komórki i aksonu. Dendryty odbierają sygnały od innych neuronów i przekazują je do ciała komórki, które integruje te sygnały. Akson, z kolei, przesyła sygnały elektryczne do innych neuronów lub komórek docelowych.

Ważnym procesem zachodzącym w neuronie jest pompa sodowo-potasowa. Ta specjalna pompa bierze udział w utrzymaniu gradientu jonowego wewnątrz i na zewnątrz komórki. Wpompuje ona aktywnie jony potasu do wnętrza komórki, jednocześnie usuwając jony sodu na zewnątrz. Ten gradient jonowy jest kluczowy dla powstawania potencjału spoczynkowego neuronu.

Przewodnictwo elektryczne jest procesem, w którym sygnał elektryczny jest przekazywany wzdłuż aksonu. Gdy neuron jest pobudzony, depolaryzacja membrany komórkowej powoduje otwarcie kanałów jonowych, co powoduje napływ jonów sodu do komórki i generuje potencjał czynnościowy. Ten potencjał czynnościowy następnie przemieszcza się wzdłuż aksonu, napędzany zmianami w przepuszczalności jonowej i repolaryzacją.

Wnioskując, zrozumienie działania neuronu, pompy sodowo-potasowej i przewodnictwa elektrycznego jest kluczowe dla poznania komunikacji międzykomórkowej w naszym układzie nerwowym. To niezwykle skomplikowany, ale fascynujący proces, który umożliwia nam odbieranie, przetwarzanie i reagowanie na bodźce zewnętrzne, co jest fundamentem naszej zdolności do myślenia, uczenia się i funkcjonowania jako istoty ludzkiej..

1. **Gry poważne teoria**

Gry poważne to rodzaj gier które mają na celu coś więcej niż czytą rozrywkę,. Ich zadaiem jest dosyarzenie edykacji lub pomoc w rowziwazywaniu problemów. Są to narzędzia łączće w sobie lementry gier z walorami educkaycjnymi, informacyjnymi lub szkoelniowymi. CHarakteryują się one kilkoma cechami:

Według Johana Huizingi, gra jest aktywnością, któ®a ma charakter wolny, odizolowany od codziennych obowiązkó, w której uczestnicy działają na podstawie ustalonych zasad w celu osiągnięci aokreślonego celu. Holender zwraca uwagę na to żę gra sama w sobie jest sytuacją mającą na celu dostarczenie rozrywki ale także budowanie więzi społecznych. W książce „Homo Ludens: o znaczeniu gry w kulturze” prezentuje wiele aspektów gdy, takich jak swoboda: ograniczony czas i przestrzeń, dobrowolne uczestnictwo. Huizinga w swojej definicji skupia się na tym, że jest to forma odrębnej aktywności, oderwanej od codziennej aktywnośći społecznej, w której uczestnicy godzą się na pewne zasady i rywalizują w określonych ramach. Gra ma charakter wolny i dobrowolny w przeciwieństwie do codziennych czynności wykonywanych przez każdego z nas.

David Michael, Sande Chen w książce “Serious Games: Games that Educate, Train, and Inform” podają definicję gry poważnej jako interaktywnej aplikacji, posiadającej zasady gry, która ma za zadanie dostarczenie wartości innej niż czysta rozrywka. Tą wartością możę być edukacja, szkolenie, lub kwestia informacyjna w formie któ®a jest bardziej przystępna dla użytkownika niż tradycyjna forma tesktsowa. Głónym celem takiej gry jest przekazanie określonych treśći użytkownikowi. Kluczowymi elementami gry poważnej są przede wsyztskm: mechanizmy standarowej gry, czyli posiadanie zasad, pewnego celu, rodzaju wyzwania i ma to na celu wzbudzenie w graczu zaangażowania w grę. Kolejnym elementem jest wspomniana wcześniej wartość edukacyjna lub informacyjna lub szkoleniowa, któ®a ma zostać w trakcie gry w pewien sposó zaspokojona.

W książce "Serious Games: Mechanisms and Effects"  autorstwa Ute Ritterfeld, Michaela Cody'ego i Petera Vordere, możemy znaleźc podobą definicję gry poważnej, mówiącą o tym żę gra poważna to gra która ma na celu dostarczenie rozrywki jednocześnie stawiając przed graczem pewne zadania, któ®a mają na celu przekazać pewne określone „poważne” treści, czyli treści edukacyjne, informacyujne, szkoleniowe. Według autorów gra poważna powinna spełniać kolka krytweiów: aspekt rozrywkowy któ®y powinien pryciągać i zachęcać użytkowników do grania (ale w przeciwieńśtwei do tradycyjnych gier, gra poważna ma założony cel inny niż rozrywka). Gra poważna powinna wykorzystywać mechanizmy grytakie jak reguły, nagroda, wyzwanie, rywalizacja, interaktywność co pozwala wciągnąć gracza w naukę.

Możę tu coś dopisać z krótkiego paperu, któ®y mam w folderze z prezentacjami na semiarium

1. **Użyte narzędzia**
   1. Unity

Silnik Unity jest jednym z najbardziej rozpowszechnionych i uniwersalnych narzędzi do tworzenia gier. Napisać kto go tworzy, kiedy został stworzony itd. Szczególną popularność posiada w branży gier komputerowych, czego dowodem jest jego zastosowanie w eilu projektach komercyjnych. Jego znaczenie wynika z wszechstronnośći, ale też innowacyjnych funkcjoalnosći, wspraraaiw dla wielu pratorm, latwosci używania (do czego przyczynia się też ogromne wsparcie społćzeniscuzytkownikow, którzy tworzą dobrowolnie bardzo dużo treści o charakterze szkoleniowym, któ®e pomaają nowym użytkownikom tego silnika na łatwiejsze wejście do tej technologii, a bardziej zawansowanym użyknoikom oferują pomoc w rozwiązywaniu problemów). Wszechstornność silnika unity objawia się w tym ze można w nim tworzyć gry na różne platformy taki ejak komputery osobiste, smartwfony, konsole lub nawet gry w wirtualnej rzecywitoi lub rozszerzonej rzeczyiwstoci. Elastyczność tego silnika przejawia się również w mozłoiwośći importowania własnych assetów lub wtyczek, lub wykorzystywaia tego typu materiałów któ®e są publiczne deostępne na zewnatrznych stronach internetowych.

Tak jak wspomniano wcześniej, łątwośc użytkowania silnika unity wynika nie tylko z intuicyjnego interfejsu, ale przede wsyztskim bardzo bohatej dokumentacji i wsparciu społeczności użytkowników. Wsparcie społeczności przejawia się w ogromnej liczbie materiałów na forach, filmó na platformach wideo, na któ®ych można zdobyć wiedzę bez względu na to na jakim poziomie zaawansowania się jest. Poczatkującym użytkownikom zostaje obniżony próg wejścia do silnika przez wzgląda na bardzo duż ą liczbe filmo instruktazowych przybliżających podstaowe zagadnienia, natomiast bardziej doswiadzceni użytkownicy, tworzący zaawansowane proejkty, jeżeli natrafią na jakiś porblem związany z silnikiem, mogą go bardzo szybko i skutecznie rozwaiżac na forach. Jest to nieoceniona przeawga tego silnika, któ®a została wytworzona organicznie przez użytkowników, a któ®ej brak przyczyniłby się być może do duoz mniejszej popularności silnika, ale przede wszystkim do dużo kniejszego komfrtu użytkowania. Sam silnik może również być udoskalany przez konakt ze swoją społęcznośćia.

Silnik unity oferuje różne zaawansowane narzędzia i funkcje któ®e sprawiają z e tworzenie gier jest prstrzze. Dotyczy to podstawowych zagadnie  jak i bardziej skompikowanych procesów.

Silnik ten posiada również pewne wady takie jka na przykład wydajność. Silnik Unity jest wymagający jeśli chodzi o zasoby komputera, szczególnie w przypadku bardziej skompikowanych gier. Możeto proawdzic do koneicznosci optymializacji grylub zwiększenia zasobów sprzętowych. Kolejną wadą jest dość spora liczba oraz rozmiar generowanych plików. Kolejną wadą jest to ze przej???

5.1.2 Inne narzędzia

Unreal Engine, stworzony przez Epic Games, jest jednym z najpopularniejszych i najpotężniejszych silników do tworzenia gier wideo oraz wirtualnych środowisk. Charakteryzuje się szeregiem zalet, które przyczyniły się do jego ogromnej popularności, ale ma również pewne wady, które warto wziąć pod uwagę.

Jedną z głównych zalet Unreal Engine jest jego ogromna siła i elastyczność. Silnik ten oferuje zaawansowane narzędzia do projektowania grafiki, tworzenia realistycznych efektów wizualnych i dynamicznej symulacji fizycznej. Wspiera również wiele platform, w tym komputery PC, konsole, urządzenia mobilne oraz wirtualną rzeczywistość.

Kolejną zaletą Unreal Engine jest jego rozbudowana społeczność. Twórcy gier mają dostęp do szerokiej bazy wiedzy, dokumentacji i darmowych zasobów udostępnianych przez społeczność użytkowników Unreal Engine. Dodatkowo, silnik ten oferuje również dedykowane narzędzia do tworzenia multiplayera oraz łatwą integrację z platformami społecznościowymi.

Niemniej jednak, Unreal Engine ma również pewne wady. Jedną z nich jest stroma krzywa nauki. Tworzenie zaawansowanych projektów w Unreal Engine może wymagać czasu i wysiłku, szczególnie dla osób, które dopiero zaczynają swoją przygodę z tworzeniem gier.

Dodatkowo, niektórzy twórcy mogą być zaniepokojeni kosztami licencji Unreal Engine. Podczas gdy korzystanie z silnika jest bezpłatne dla projektów, które generują roczne przychody poniżej pewnego progu, większe i bardziej komercyjne projekty mogą wymagać opłat licencyjnych.

Podsumowując, Unreal Engine jest potężnym narzędziem do tworzenia gier i wirtualnych środowisk, oferującym ogromne możliwości i rozbudowaną społeczność. Jednak, z uwagi na stromą krzywą nauki i potencjalne koszty licencji, należy dokładnie rozważyć te czynniki przed podjęciem decyzji o wykorzystaniu Unreal Engine w swoim projekcie.

Godot engine – napisac cos

Godot Engine to darmowy i otwartoźródłowy silnik do tworzenia gier, który zdobywa coraz większą popularność wśród twórców gier. Posiada wiele zalet, które przyczyniają się do jego rosnącej popularności, ale także pewne wady, które warto wziąć pod uwagę.

Jedną z głównych zalet Godot Engine jest jego bezpłatność i otwartoźródłowy charakter. Może być używany do tworzenia gier na różne platformy, takie jak komputery PC, konsole, urządzenia mobilne i wiele innych. Jako otwarte oprogramowanie, Godot Engine daje użytkownikom pełną kontrolę nad kodem źródłowym i możliwość dostosowania go do swoich potrzeb.

Kolejną zaletą Godot Engine jest jego intuicyjny interfejs i łatwość nauki. Silnik ten oferuje przyjazne dla użytkownika narzędzia do projektowania i tworzenia gier. Posiada również bogatą dokumentację i aktywną społeczność, która dzieli się wiedzą i udostępnia darmowe zasoby.

Godot Engine wspiera również tworzenie gier 2D i 3D, oferując zaawansowane funkcje i efekty wizualne. Silnik ten posiada również wiele wbudowanych narzędzi do zarządzania zasobami, animacji, fizyki, dźwięku i wiele innych, co ułatwia proces tworzenia gier.

Wśród wad Godot Engine można wymienić mniejszą popularność w porównaniu z niektórymi innymi silnikami. To oznacza, że może być trudniej znaleźć wsparcie i pomoc w przypadku bardziej zaawansowanych problemów. Ponadto, niektóre funkcje mogą nie być tak rozbudowane jak w niektórych innych silnikach, co może być ograniczeniem dla bardziej zaawansowanych projektów.

Podsumowując, Godot Engine jest bezpłatnym, otwartoźródłowym silnikiem do tworzenia gier, oferującym wiele zalet, takich jak łatwość nauki, wsparcie dla różnych platform i bogate funkcje. Jednak, z uwagi na mniejszą popularność i pewne ograniczenia, warto dokładnie rozważyć, czy spełnia on potrzeby konkretnego projektu gry.

BLENDER

Blender to potężne narzędzie do tworzenia grafiki trójwymiarowej, które zdobyło sobie uznanie wśród profesjonalistów i entuzjastów animacji i grafiki komputerowej. Jest to darmowy i otwartoźródłowy program, który oferuje niezwykle bogate możliwości tworzenia modeli, animacji, renderowania, kompozycji, tworzenia tekstur i wiele więcej.

Jedną z największych zalet Blendera jest jego wszechstronność. Można go wykorzystać do tworzenia różnorodnych projektów 3D, takich jak filmy animowane, gry wideo, wizualizacje architektoniczne, efekty specjalne czy projekty produktów. Narzędzia do modelowania pozwalają na tworzenie dowolnych kształtów i struktur, a narzędzia animacyjne umożliwiają ożywienie tych modeli za pomocą różnych technik animacji.

Blender oferuje również zaawansowane możliwości renderowania, co pozwala na generowanie wysokiej jakości obrazów i animacji. Dzięki wbudowanemu silnikowi Cycles oraz wsparciu dla renderowania GPU i CPU, użytkownicy mogą uzyskiwać foto-realisticzne efekty i kontrolować oświetlenie, tekstury, materiały i wiele innych parametrów.

Niestety, Blender nie jest pozbawiony wad. Ze względu na swoje zaawansowane możliwości, nauka obsługi programu może być czasochłonna i wymagać pewnego wysiłku. Interfejs użytkownika może wydawać się skomplikowany dla początkujących, jednak istnieje wiele darmowych tutoriali i materiałów edukacyjnych, które mogą pomóc w opanowaniu Blendera.

Podsumowując, Blender to niezwykle potężne narzędzie do tworzenia grafiki trójwymiarowej, które oferuje szeroki zakres możliwości dla profesjonalistów i entuzjastów. Jego wszechstronność, darmowa dostępność i aktywna społeczność użytkowników czynią go atrakcyjnym wyborem dla tych, którzy chcą rozwijać się w dziedzinie tworzenia grafiki 3D. Mimo pewnych trudności z nauką obsługi programu, Blender stanowi doskonałą platformę dla kreatywnych projektów i eksploracji wirtualnego świata trójwymiarowego.

1. **Opis implementacji**
   1. Zarys ogólny

A screenshot of a computer

Description automatically generated with low confidence

***InputHandler.cs***

S

***Menu.cs***

Jest to skrypt służący do przemieszczania się między kolejnymi scenami w grze. Zawiera on proste metody, zmieniające obecną scenę.

**public** **void** nextScreen()

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);

}

Metoda *nextScreen()* przełącza obecną scenę, na kolejną za pomocą argumentu *SceneManager.GetActiveScene().buildIndex + 1,* natomiastmetoda ta przyjmuje jako argument liczbę całkowitą, dzięki czemu możliwe jest przełączenie się do konkretnej sceny.

***ZoomMouse.cs***

Jest to skrypt, służący do przybliżania i oddalania modelu 3D za pomocą myszy. Zmienne zoomSpeed, maxZoomOut, maxZoomIn służą do kontrolowania prostych parametrów związanych z prędkością przybliżania, maksymalnego przybliżenia oraz oddalenia modelu od kamery w scenie.

**public** **class** ZoomMouseWheel : MonoBehaviour

{

**private** **float** zoomSpeed = 0.5f;

**private** **float** maxZoomOut = 0.5f;

**private** **float** maxZoomIn = 1.5f;

**private** **float** zoom;

**void** Update()

    {

        zoom = Input.GetAxis("Mouse ScrollWheel");

        transform.localScale += **new** Vector3(zoom, zoom, zoom) \* zoomSpeed;

**if**(transform.localScale.x < maxZoomOut)

        {

            transform.localScale = **new** Vector3(maxZoomOut, maxZoomOut, maxZoomOut);

        }

**else** **if** (transform.localScale.x > maxZoomIn)

        {

            transform.localScale = **new** Vector3(maxZoomIn, maxZoomIn, maxZoomIn);

        }

    }

}

W metodzie Update(), znajdują się dwie instrukcje warunkowe *if()*, których zadaniem jest uniemożliwienie oddalenia lub przybliżenia powyżej zadeklarowanej wartości zmiennej *maxZoomOut* i *maxZoomIn*.

***MouseHover.cs***

Jest to skrypt, który pozwala użytkownikowi na kolorowe podświetlenie wybranego obszaru modelu 3D, za pomocą najechania na ten obszar kursorem myszy.

**public** **class** MouseHover : MonoBehaviour

{

**private** Color newSphereColor;

**private** GameObject sphere;

**private** Renderer sphereRenderer;

**void** Start()

    {

        sphereRenderer = sphere.GetComponent<Renderer>();

    }

**private** **void** OnMouseOver()

    {

**if** (GrabRotation.isClickingTurnedOn)

        {

            activateColor();

        }

    }

**private** **void** OnMouseExit()

    {

**if** (GrabRotation.isClickingTurnedOn)

        {

            defaultColor();

        }

    }

**public** **void** activateColor()

    {

        sphereRenderer.material.SetColor("\_Color", newSphereColor);

    }

**public** **void** defaultColor()

    {

        newSphereColor = **new** Color(1.0f, 1.0f, 1.0f, 0.0f);

        sphereRenderer.material.SetColor("\_Color", newSphereColor);

    }

}

Metoda *OnMouseOver()* służy do obsługi zdarzenia w którym użytkownik „najechał” kursorem myszy na wybrany fragment modelu. Znajdująca się wewnątrz instrukcja warunkowa *if()* ma za zadanie manipulować podświetlaniem, i wyłączać tą opcję za pomocą zmiennej *GrabRotation.isClickingTurnedOn*. Zmienna ta jest statyczna i jej wartość może być zmieniona w innym skrypcie. Metoda *OnMouseExit()* działa bardzo analogicznie, natomiast obsługuje zdarzenie w którym kursor myszy opuścił podświetlany obszar.

Metoda *activateColor()* ma za zadanie zmienić kolor wybranego fragmentu modelu, tak aby był on wizualnie wyróżniony. Metoda *defaultColor()*, służy do przywrócenia oryginalnego koloru fragmentowi modelu. Jest tu mowa o fragmencie modelu, ponieważ intencją tego skryptu było podświetlenie wybranego fragmentu, a nie całego modelu (czyli na przykład podświetlenie jednego płata mózgu).

***BrainRotation.cs***

Jest to skrypt, którego zadaniem jest obracanie całego modelu mózgu.

**public** **class** BrainRotation : MonoBehaviour

{

**public** **float** rotationSpeed = 1000f;

**public** **bool** isBrainRotating = **false**;

**public** Transform centerOfRotation;

**void** Update()

    {

**if** (isBrainRotating)

        {

**float** mouseX = Input.GetAxis("Mouse X");

**float** mouseY = Input.GetAxis("Mouse Y");

            transform.RotateAround(centerOfRotation.position, Vector3.down, mouseX \* rotationSpeed \* Time.deltaTime);

            transform.RotateAround(centerOfRotation.position, Vector3.right, mouseY \* rotationSpeed \* Time.deltaTime);

        }

        isBrainRotating = **false**;

    }

}

Zmienna *isBrainRotating* jest flagą, której ustawienie na wartość *true* w skrypcie *GrabRotation.cs*, przekazuje informację o tym, że model może się obracać. Jest ona istotna z tego względu, że jej brak powodowałby obracanie się modelu cały czas, zgodnie z ruchem myszy.

***GrabRotation.cs***

Jest to skrypt, którego zadaniem jest obsługa zdarzenia, gdzie użytkownik przytrzymuje lewy przycisk myszy na danym fragmencie modelu. Szczególnie ważne jest tutaj rozróżnienie sytuacji w której nad danym fragmentem modelu następuje naciśnięcie myszy, a naciśnięcie i przytrzymanie. W przypadku „naciśnięcia i przytrzymania” użytkownik może obracać model, a w przypadku zwykłego „kliknięcia” następuje przejście do trybu tekstowego, w którym użytkownik może przeczytać informacje tekstowe na temat wybranego fragmentu.

**public** **class** GrabRotation : MonoBehaviour

{

**public** BrainRotation brainRotation;

**public** MoveAnimation moveAnimation;

**private** **int** isDragCounter = 0;

**private** **const** **int** dragTrigger = 15;

**public** **bool** leftSideInactiveMode = **false**;

**public** **static** **bool** isClickingTurnedOn = **true**;

**private** **void** Start()

    {

        isClickingTurnedOn = **true**;

        leftSideInactiveMode = **false**;

    }

**void** OnMouseDrag()

    {

**if** (isClickingTurnedOn)

        {

            isDragCounter++;

**if** (isDragCounter > dragTrigger)

            {

                brainRotation.isBrainRotating = **true**;

            }

        }

    }

**private** **void** OnMouseUp()

    {

**if** (isClickingTurnedOn)

        {

**if** (isDragCounter <= dragTrigger)

            {

                moveAnimation.enableSideLeftAnimation();

            }

            isDragCounter = 0;

        }

    }

}

W skrypcie używane są metody *OnMouseDrag()* oraz *OnMouseUp()*, których zadaniem jest obsługa naciśnięcia przycisku myszy nad danym obszarem oraz zwolnienie tego przycisku.

W metodzie *OnMouseDrag()*, znajduje się mechanizm rozróżnienia naciśnięcia przycisku myszy, oraz naciśnięcia i przytrzymania. Możliwe jest to dzięki licznikowi *isDragCounter*, który rozpoczyna inkrementację w momencie naciśnięcia myszy. Jeżeli użytkownik przestał naciskać przycisk przed osiągnięciem pewnego *dragTrigger* (który jest stałą zmienną typu *int*), to uznawane jest to za „kliknięcie” i flaga *isBrainRotating* pozostaje w stanie false. W przeciwnym przypadku, gdy wartość *isDragCounter* przekroczyła progową wartość *dragTrigger*, uznawane jest to za „naciśnięcie i przytrzymanie”  a wartość flagi *isBrainRotating* jest zmieniana na *true*, co jest istotną informacją dla skryptu *BrainRotating.cs.*

Metoda *OnMouseUp()* uruchamia się w momencie puszczenia przycisku myszy. Sprawdzane wtedy jest za pomocą instrukcji warunkowej *if()* czy użytkownik „kliknął” czy „kliknął i przytrzymał” fragment modelu. W przypadku pierwszej sytuacji, uruchamiana jest metoda *moveAnimation.enableSideLeftAnimation()* której zadaniem jest przełączenie sceny do trybu wyświetlania tekstu.

***MoveAnimation.cs***

Jest to skrypt, którego zadaniem jest przeniesienie sceny z trybu widoku, do trybu tekstowego. Jest uruchamiany w momencie gdy użytkownik przyciskiem myszy naciśnie na wybrany fragment modelu. Tryb tekstowy charakteryzuje się tym, że wybrany fragment mózgu zostaje podświetlony, cały model mózgu jest nieaktywny (czyli niemożliwe jest jego obracanie i przybliżanie) a obok zostaje zaprezentowany opis dotyczący tego konkretnego fragmentu mózgu. Po przeczytaniu tekstu, użytkownik ma możliwość wrócenia do trybu widoku (służy do tego przycisk „wyśrodkuj”) lub przejścia do kolejnej sceny (służy do tego przycisk „wejdź głębiej”).

**public** **class** MoveAnimation : MonoBehaviour

{

**public** **bool** isSideLeftAnimationOn = **false**;

**public** Transform brainPositionForDisplayingText;

**public** Transform cameraCentralPoint;

**public** Transform brainContainer;

**private** **float** movementSpeed = 6.5f;

**private** Vector3 vectorCentreOfLobeToCentreOfBrain;

**public** BrainDescriptionText brainDescriptionText;

**public** MouseHover mouseHover;

**public** GameObject centralizeBrainButton;

**public** GameObject goInsideButton;

    Vector3 brainScaleSave;

**private** **void** Start()

    {

        brainScaleSave = brainContainer.transform.localScale;

        setTextVisibility(**false**);

    }

**void** Update()

    {

        vectorCentreOfLobeToCentreOfBrain = brainContainer.transform.position - transform.position;

**if** (isSideLeftAnimationOn)

        {

            moveBrainLeft();

        }

**if** (BrainCentralization.isCentralizationMoveOn)

        {

            centralizationAdditionalActions();

            GrabRotation.isClickingTurnedOn = **true**;

        }

    }

**public** **void** setTextVisibility(**bool** isTextVisible)

    {

**if** (isTextVisible)

        {

            brainDescriptionText.textContainer.gameObject.SetActive(**true**);

            centralizeBrainButton.gameObject.SetActive(**true**);

            goInsideButton.gameObject.SetActive(**true**);

        }

**else**

        {

            brainDescriptionText.textContainer.gameObject.SetActive(**false**);

            centralizeBrainButton.gameObject.SetActive(**false**);

            goInsideButton.gameObject.SetActive(**false**);

        }

    }

**void** moveBrainLeft()

    {

        brainContainer.transform.position = Vector3.Lerp(

                brainContainer.transform.position,

                brainPositionForDisplayingText.transform.position + vectorCentreOfLobeToCentreOfBrain,

                Time.deltaTime \* movementSpeed);

**if** ((brainPositionForDisplayingText.transform.position.x + vectorCentreOfLobeToCentreOfBrain.x)

            - brainContainer.transform.position.x < 0.3f)

        {

            mouseHover.activateColor();

            setTextVisibility(**true**);

            scaleUpBrain();

            GrabRotation.isClickingTurnedOn = **false**;

        }

    }

**public** **void** enableSideLeftAnimation()

    {

        isSideLeftAnimationOn = **true**;

    }

**public** **void** disableSideLeftAnimation()

    {

        isSideLeftAnimationOn = **false**;

        setTextVisibility(**false**);

    }

**void** centralizationAdditionalActions()

    {

        disableSideLeftAnimation();

        scaleDownBrain();

        mouseHover.defaultColor();

    }

**void** scaleUpBrain()

    {

        brainContainer.transform.localScale = Vector3.Lerp(brainContainer.transform.localScale, brainContainer.transform.localScale \* 5.0f, 0.2f \* Time.deltaTime);

    }

**void** scaleDownBrain()

    {

        brainContainer.transform.localScale = brainScaleSave;

    }

}

Skrypt w metodzie *Start()* ustawia początkowe wartości, a metodą *Update()* aktualizuje wydarzenia na scenie. Zadaniem zmiennej *vectorCentreOfLobeToCentreOfBrain* jest przechwycenie pozycji odpowiednich punktów na scenie, dzięki czemu możliwa jest sytuacja w której po przejściu do trybu tekstowego, model przybliżony mózgu zostaje wyśrodkowany w stosunku do punktu znajdującego się w wybranym przez użytkownika fragmencie mózgu.

Metoda *setTextVisibility()* ma za zadanie wyświetlić boczny tekst, oraz odpowiednie przyciski.

Metoda *moveBrainLeft()* ma za zadanie przenieść w odpowiednim czasie cały model mózgu z punktu określonego zmienną *brainContainer.transform.position* do punktu *brainPositionForDisplayingText.transform.position*, dodatkowo przesuniętego o *vectorCentreOfLobeToCentreOfBrain.* Przesunięciedokonuje się za pomocą metody *Vector3.Lerp,* która wykonuje liniowąinterpolację między punktem początkowym a końcowym. Po przesunięciu następuje instrukcja warunkowa *if()* sprawdzająca czy model mózgu przesunął się na odpowiednią odległość. Jeśli tak się stało, to wybrany fragment mózgu podświetlany jest na wyróżniający kolor, wyświetlany jest boczny tekst, model mózgu jest odpowiednio powiększany a możliwość „klikania” w obszary mózgu zostaje tymczasowo wyłączona.

Kolejną istotną metodą jest *centralizationAdditionalActions()*, która wykonuje się w momencie gdy użytkownik nacisnął przycisk „wyśrodkuj”. Zmniejsza ona odpowiednio model mózgu, przywraca domyślny kolor wybranego fragmentu mózgu oraz umożliwia powrót do trybu widoku.

***BrainCentralization.cs***

Jest to skrypt, którego zadaniem jest przełączenie sceny z trybu tekstowego do trybu widoku. Towarzyszy temu usunięcie ze sceny opisu fragmentu mózgu, odpowiednich przycisków oraz przesunięcie modelu mózgu z powrotem do punktu centralnego sceny.

**public** **class** BrainCentralization : MonoBehaviour

{

**public** **static** **bool** isCentralizationMoveOn = **false**;

**public** Transform cameraCentralPoint;

**private** **float** movementSpeed = 6.5f;

**private** **const** **float** distanceBrainToCentreTriggeringFlag = 0.3f;

**void** Update()

    {

**if** (isCentralizationMoveOn)

        {

            brainCentralizationMovement();

        }

**if** (cameraCentralPoint.transform.position.x - transform.position.x < distanceBrainToCentreTriggeringFlag)

        {

            isCentralizationMoveOn = **false**;

        }

    }

**public** **void** tickCentralizationFLag()

    {

            isCentralizationMoveOn = **true**;

    }

**public** **void** brainCentralizationMovement()

    {

        transform.position = Vector3.Lerp(

            transform.position,

            cameraCentralPoint.transform.position,

            Time.deltaTime \* movementSpeed);

    }

}

Metoda *brainCentralizationMovement()* do przesunięcia modelu mózgu na środek sceny używa opisywanej wcześniej metody *Vector3.Lerp*.

A picture containing screenshot, text, operating system, font

Description automatically generated

Gra rozpoczyna się od pobrania od użytkownika imienia.

Imię pobierane jest za pomocą skryptu *InputHandler.cs*.

A screenshot of a computer screen

Description automatically generated with low confidence

Następnie gracz przechodzi do ekranu w którym może wybrać czy chce przejść do trybu widoku budowy mózgu lub budowy neuronu.

A picture containing screenshot, brain

Description automatically generated

W przypadku budowy mózgu, tryb widoku podzielony jest na 3 etapy:

- półkule mózgu,

- płaty mózgu,

- przekrój mózgu,

A green and pink brain

Description automatically generated with medium confidence

A green and pink brain

Description automatically generated with low confidence

W każdym z tych etapów, użytkownik może wykonać te same akcje, czyli: przybliżanie i oddalanie widoku modelu 3D mózgu, obracanie modelu, zazaczenie wybrango fragmentu i wyświetlenie opisu dotyczącego danego fragmentu (tryb tekstowy). Następnie użytkownik może wybrać, czy powraca do trybu widoku, czy chce przejść do kolejnego etapu. Wyjątkiem jest ostatni etap (widok przekroju mózgu), po którym gracz ma do wyboru, powrócić do ekranu budowy mózgu, przejść do budpwy neuronu, lub przejść do testu wiedzy.

A screenshot of a computer screen

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with medium confidence

6.2 Tryb modelu

6.2.1 Przybliżanie/oddalanie i obracanie

6.2.2 Zaznaczanie konkretnego obszaru

6.2.2 Tryb opisu

6.3 Test wiedzy

6.4 Tabela wyników

6.5 Zapisywanie danych do pliku

1. **Tutorial dla użytkownika**
2. **Anketa**

Ankieta

Opis badania:

Głównym celem tego badania jest przeprowadzenie szczegółowej analizy opinii młodych osób na temat gier poważnych jako formy nauki. Badanie zostanie przeprowadzone za pomocą starannie opracowanej ankiety w formie Google Forms, skierowanej do grupy młodych ankietowanych. Ankieta będzie dostępna online, co umożliwi łatwe zbieranie danych i dogłębną analizę wyników.

Ankieta będzie zawierać różnorodne pytania dotyczące znajomości gier poważnych. Przykładowe pytania mogą obejmować: "Czy jesteś zaznajomiony z pojęciem 'gry poważne'?", "Czy kiedykolwiek miałeś/miałaś okazję zagrać w gry poważne?", "W jakim stopniu korzystasz z gier poważnych w celach edukacyjnych?" oraz "Czy uważasz, że gry poważne stanowią skuteczną formę nauki?". Odpowiedzi będą udostępnione w formie wyboru jednokrotnego lub wielokrotnego.

Wnioski z wyników ankiety:

Dogłębna analiza odpowiedzi na ankietę pozwoli na wyciągnięcie istotnych wniosków. Na podstawie wyników można stwierdzić, że gry poważne są powszechnie znane wśród ankietowanych młodych osób, ponieważ połowa z nich ma z nimi styczność. Co więcej, zdumiewające jest również to, że znaczna część ankietowanych korzysta z gier poważnych w celach edukacyjnych. Warto zauważyć, że ankietowani sięgają po gry poważne w sytuacjach, gdy są zainteresowani danym tematem lub chcą pogłębić swoją wiedzę w danej dziedzinie.

Ponadto, wyniki wskazują, że gry poważne są postrzegane jako nie tylko interesująca forma nauki, ale również jako skuteczne narzędzie edukacyjne. Ankietowani podkreślają, że gry poważne angażują ich uwagę, rozwijają umiejętność logicznego myślenia, pomagają w rozwiązywaniu problemów, a także wspierają pamięć i koncentrację. Wielu z ankietowanych zwraca również uwagę na aspekty społeczne gier poważnych, takie jak współpraca i komunikacja w zespołach, które sprzyjają nabywaniu umiejętności interpersonalnych.

Jednocześnie, połowa ankietowanych nie jest zaznajomiona z grami poważnymi, co sugeruje, że istnieje potrzeba większego promowania i edukowania na temat tego rodzaju gier. Wniosek płynący z badania wskazuje na konieczność podjęcia działań mających na celu zwiększenie świadomości młodych osób na temat możliwości i korzyści płynących z gier poważnych jako narzędzia edukacyjnego. Należy tworzyć kampanie informacyjne, organizować seminaria i warsztaty, które zwiększą świadomość młodzieży na temat potencjału edukacyjnego gier poważnych.

Podsumowując, przeprowadzone badanie potwierdza, że gry poważne mają potencjał jako skuteczna forma nauki i cieszą się pewnym stopniem popularności wśród młodych osób. Jednakże, istnieje potrzeba dalszego rozwoju i promocji tych gier w celu zapewnienia większego dostępu do nich i zwiększenia ich wykorzystania jako narzędzia edukacyjnego. Warto inwestować w edukację na temat gier poważnych, aby młode pokolenie mogło korzystać z pełnego potencjału tych innowacyjnych i angażujących form nauki.

1. **Podsumowanie**

Podsumowanie i zakończenie:

Edukacyjna gra poważna, która porusza tematykę funkcjonowania ludzkiego mózgu oraz budowy neuronu, jest niezwykle wartościowym i fascynującym narzędziem edukacyjnym. Przedstawione w grze informacje na temat mózgu, jego funkcji oraz budowy neuronu, mogą dostarczyć uczącym się niezwykłej wiedzy na temat tego niezwykłego organu i jego fundamentalnych elementów.

Gry poważne pozwalają na eksplorację i doświadczanie procesów poznawczych, funkcji mózgu oraz działania neuronów w sposób interaktywny, angażujący i przyjemny. Gracze mają szansę zgłębiać tematy związane z percepcją, pamięcią, myśleniem, emocjami, co przyczynia się do lepszego zrozumienia działania ich własnego umysłu.

Jedną z najważniejszych zalet edukacyjnych gier poważnych poruszających tematykę mózgu i neuronów jest to, że uczący się stają się aktywnymi uczestnikami w procesie poznawczym. Poprzez podejmowanie decyzji, eksperymentowanie i obserwowanie skutków swoich działań, mogą samodzielnie odkrywać zasady funkcjonowania mózgu oraz rozwijać swoje umiejętności i wiedzę. To dynamiczne podejście do nauki umożliwia rozwijanie kreatywności, umiejętności rozwiązywania problemów, a także poszerzanie wiedzy w sposób interaktywny i angażujący.

Edukacyjne gry poważne, skupiające się na funkcjonowaniu mózgu i budowie neuronu, to doskonały sposób na rozwijanie wiedzy i zrozumienia tych zagadnień. Dają uczącym się możliwość pogłębiania wiedzy na temat ludzkiego mózgu w sposób atrakcyjny, angażujący i dostosowany do ich indywidualnych potrzeb i umiejętności. Takie narzędzie edukacyjne może wspierać proces nauki, rozwijać umiejętności poznawcze oraz budować ciekawość świata.

Jednym z kluczowych wniosków płynących z edukacyjnych gier poważnych dotyczących mózgu i neuronów jest to, że takie narzędzia mogą przyczynić się do lepszego zrozumienia naszego umysłu i jego możliwości. Poprzez eksplorowanie i eksperymentowanie z różnymi aspektami mózgu, gracze mogą zdobywać wiedzę i umiejętności, które mają praktyczne zastosowanie w życiu codziennym.

Dodatkowo, takie gry mogą również wpływać na postrzeganie nauki przez młodych ludzi. Dzięki interaktywności, angażującej rozgrywce i motywacji do osiągania postępów, uczący się mogą doświadczyć, jak fascynujące i satysfakcjonujące może być zdobywanie wiedzy i rozwijanie umiejętności. To z kolei może przyczynić się do większej motywacji i zaangażowania w proces nauki.

Podsumowując, edukacyjne gry poważne skupiające się na funkcjonowaniu mózgu i budowie neuronu stanowią niezwykle wartościowe narzędzie edukacyjne. Przy ich pomocy uczący się mogą odkrywać i zgłębiać tajemnice naszego mózgu w sposób interaktywny i angażujący. Dzięki temu mogą rozwijać umiejętności poznawcze, poszerzać wiedzę i lepiej zrozumieć potencjał swojego umysłu. Edukacyjne gry poważne mogą odgrywać kluczową rolę w wspieraniu proces

1. **Bibliografia**

d