**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Факультет ПИиКТ**

**Дисциплина:**

**«Нейроинтерфейсы информационных систем»**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4**

**Выполнили:**

Студенты гр. P33212

Курдюмов Дмитрий Александрович,

Колегова Мария Валерьевна

**Проверил:**

Штенников Дмитрий Геннадьевич

Санкт-Петербург

2021 г

Оглавление

[Введение 3](#_Toc73357660)

[Описание работы 4](#_Toc73357661)

[Код 5](#_Toc73357662)

Введение

Данная лабораторная работа предполагает написание VR-игры при помощи Unity c учетом биометричеких показаний. Для реализации данного функционала было решено использовать среду Unity для создания игровой логики и интеграции с аппаратным комплексом HTC VIVE при помощи готовых скриптов из библиотеки Steam VR.

Чтение данных пульса игрока проводилось при помощи программы, использующей возможности Universal Windows Platform, которая облегчает интеграцию различных типов устройств и может запускаться на множестве платформ, использующих ОС семейства Windows. Исходные файлы программы выложены компанией Microsoft на github, что позволило адаптировать ее под наши требования.

Для изучения вариабельности сердечного ритма игрока во время игры использовалась метрика индекса напряженности по Баевскому, показывающая степень централизации в управлении сердечным ритмом.

Для обмена данными между приложением, считывающим показания пульса, игрой и программой расчета вариабельности сердечного ритма, был создан небольшой web-сервер, работающий на localhost интерфейсе и обеспечивающий простой доступ к данным пульса в текущий момент времени для приложений, нуждающихся в нем.

Описание работы

Основная идея игры: при помощи синего и красного мечей (или 1 двуцветного меча) разрезать вылетающие кубики с учетом соответствующего цвета и направления, обозначенных цветом и расположением специальных меток на кубиках (см рис. 1).

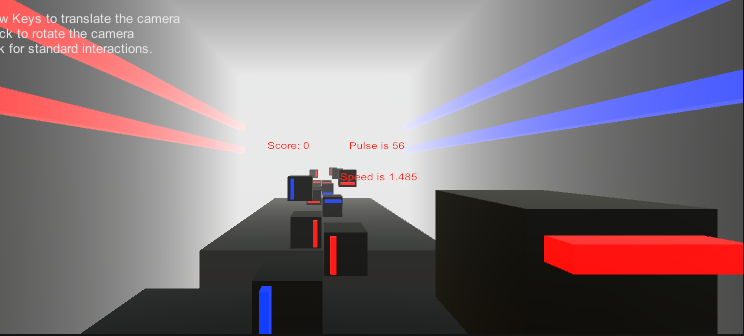


Рисунок 1. Общий вид игры

Во время игры ведется счет (1 уничтоженный кубик – 1 очко), также можно насладиться прекрасным музыкальным сопровождением (Skrillex – Go on).



Рисунок 2. Подсчет очков

Скорость кубиков находится в обратной зависимости от пульса игрока (чем выше пульс, тем меньше скорость). Данная функция регулируется двумя параметрами – максимальным пульсом, который может установить себе игрок, и максимальной скоростью кубиков. Ниже представлена такая зависимость для максимального пульса 110 уд/мин и скорости 4.

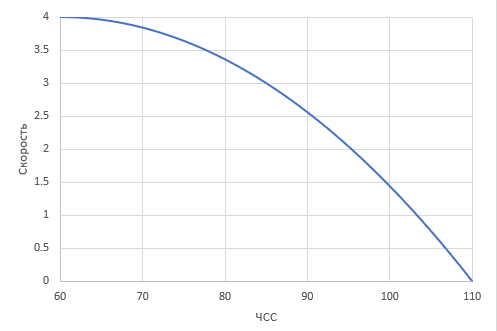


Рисунок 3. Зависимость скорости от пульса игрока

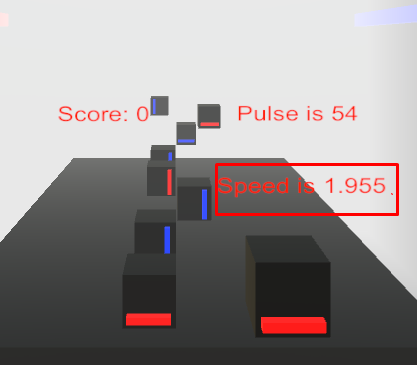
 Продемонстрируем работу игры при разных значениях пульса:

Рисунок 4. Изменение сложности игры в зависимости от пульса

Также во время игры высчитывается индекс напряженности по Баевскому. Пример работы скрипта:

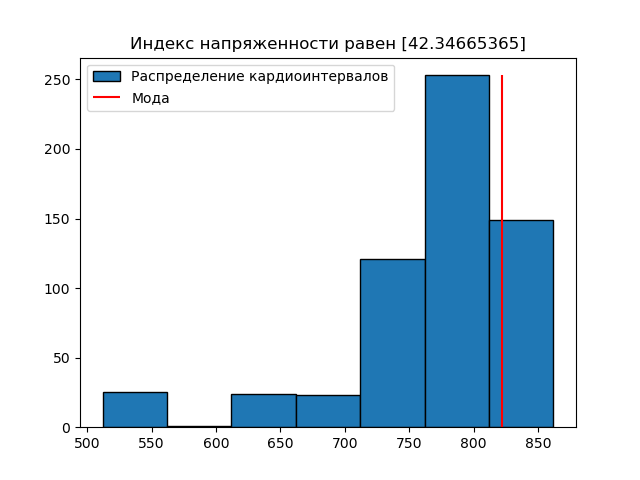


Рисунок 5.Вывод программы

Код

Данные с Bluetooth – пульсометра записываются на сервер, а затем считываются в Unity. Ниже представлен код для отправки на сервер нового значения пульса при помощи вызова метода GetStringAsync с указанием url-запроса.

try

{

newValue = newValue.Substring(newValue.IndexOf(':') + 2);

var responseString = await client.GetStringAsync($"http://127.0.0.1:8000/hr/update?bpm={newValue}");

}

В свою очередь, чтение пульса происходит при помощи следующего блока кода в среде Unity:

try

{

var resp = await client.GetStringAsync($"http://127.0.0.1:8000/hr/get");

var bpm = int.Parse(resp);

}

Следующий скриншот показывает лог запросов к серверу во время работы игры.

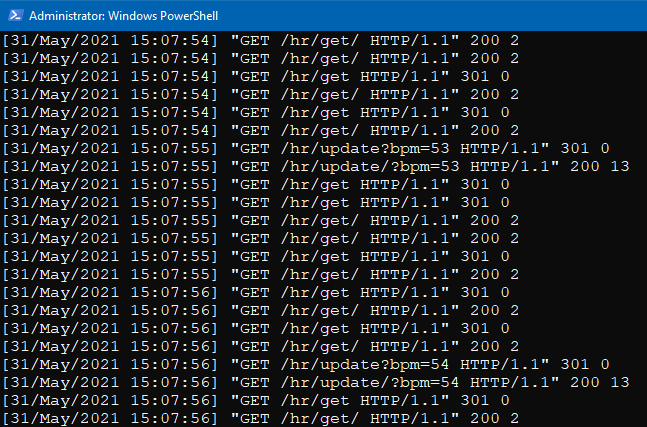


Рисунок 6. Лог http-запросов

Скорость движения кубиков имеет параболическую зависимость от значения пульса. Каждый новый кадр снимаются текущие данные пульса и вычисляется соответствующая скорость движения кубиков. До начала игры есть возможность задать максимальную сложность (скорость) и максимальный пульс игрока, от чего будет зависеть вид параболы (изменение скорости кубиков).

Далее на скриншоте представлен интерфейс настройки сложности игры в среде Unity:

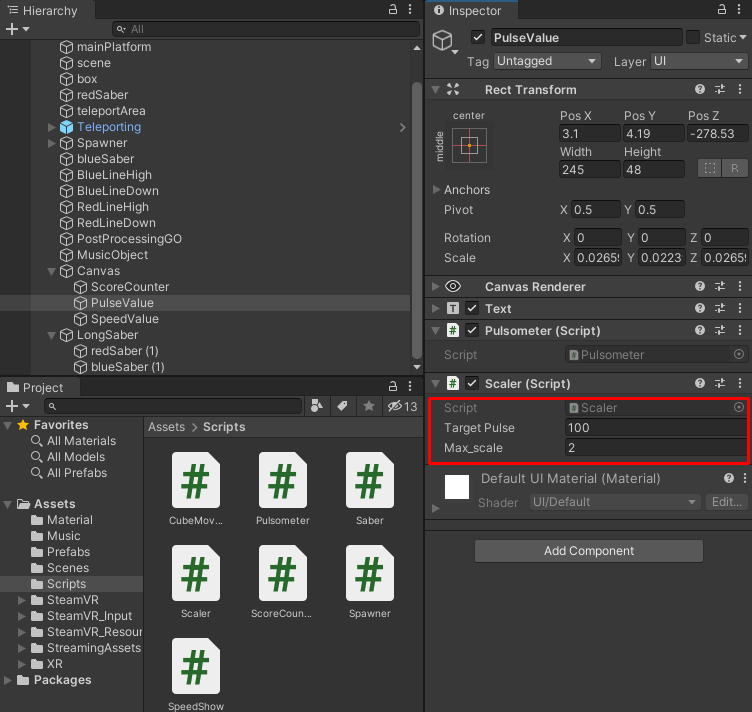


Рисунок 7. Установка значений Target Pulse и Max Scale

По этим двум параметрам представляется возможным задать семейство кривых, имеющих общий вид с продемонстрированной на рисунке 3 при помощи следующего кода:

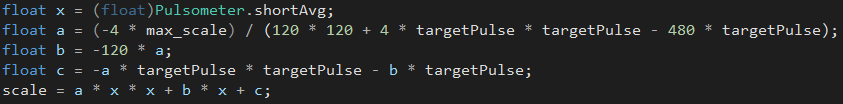


Рисунок 8. Расчет текущего множителя для скорости движения кубиков

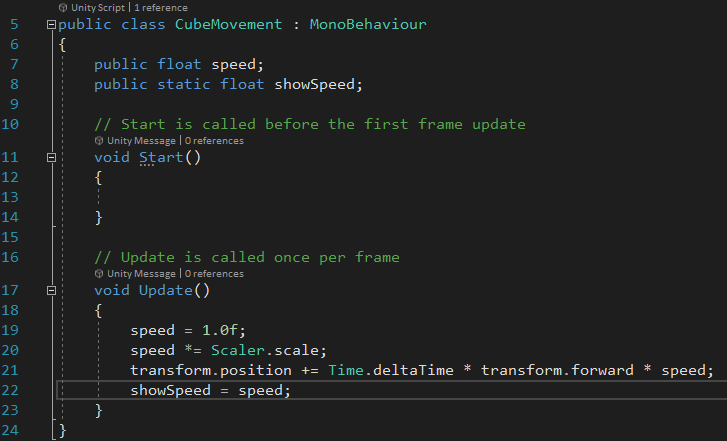


Рисунок 9. Расчет скорости с учетом полученной сложности

Наращивание счетчика результатов игры происходит в момент, когда один из мечей «разрезает» кубик соответствующего цвета в правильном направлении. После проверки соответствия цвета осуществляется вычисление угла между нормалями к кубику (верхней грани, по которой приходится удар) и к мечу (к бьющей стороне) – требуется, чтобы угол был больше 100 градусов.

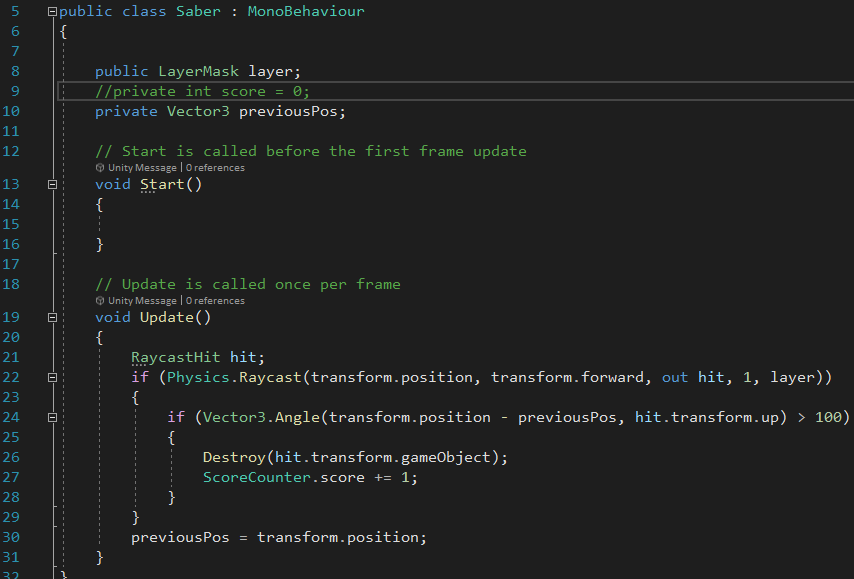


Рисунок 10. Условия удара по кубику

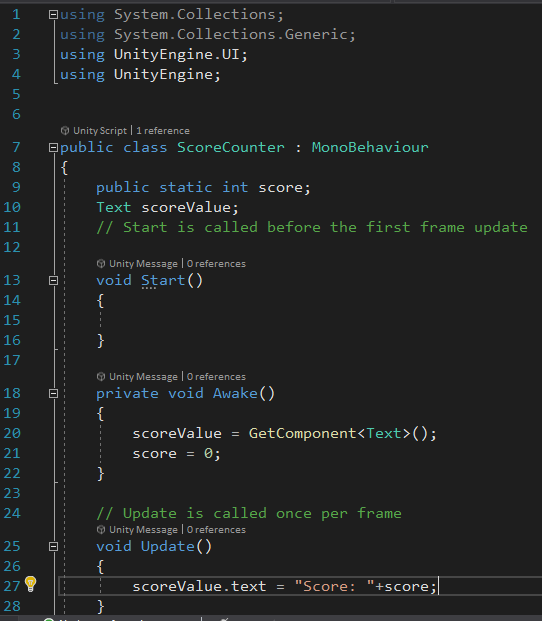


Рисунок 11. Вывод счетчика кубиков на экран

Индекс напряженности по Баевскому вычислялся по формуле: , где ИН – индекс напряженности, AMo — процент КИ, попадающих в модальный бин, dRR — разность меду максимальным и минимальным КИ, Mo - значение КИ, делящее всю выборку пополам, при симметричном распределении мода близка к среднему значению. Данные пульса записывались в файл, после чего находились кардиоинтервалы путем деления 60 на последовательное значение пульса. Далее выборка разбивалась по бинам в 50 мс, определялся наиболее часто встречаемый кардиоинтервал (мода) и количество его повторений, чтобы вычислить AMo. После определения максимального и минимального кардиоинтервалов вычислялся dRR. Вычисления выполнялись при помощи скрипта на Python.

**from** urllib**.**request **import** urlopen

**import** numpy **as** np

**import** time

num\_obs **=** 600

values **=** np**.**zeros**(**num\_obs**)**

**for** i **in** range**(**num\_obs**):**

value **=** **int(**urlopen**(**'http://127.0.0.1:8000/hr/get'**).**read**().**decode**())**

values**[**i**]** **=** value

***#print(value)***

time**.**sleep**(**1**)**

**print(**values**)**

**import** scipy**.**stats **as** stats

CI **=** **(**60**\***1000**)/**values

dRR **=** max**(**CI**)-**min**(**CI**)**

M **=** stats**.**mode**(**CI**).**mode

***#M = np.median(CI)***

**import** matplotlib**.**pyplot **as** plt

n**,** bins**,** p **=** plt**.**hist**(**CI**,** bins**=**range**(int(**min**(**CI**)),** **int(**max**(**CI**)),** 50**),** edgecolor**=**'black'**,** label**=**'Распределение кардиоинтервалов'**)**

plt**.**vlines**(**M**,** ymin**=**0**,** ymax**=**max**(**n**),** color**=**'r'**,** label**=**'Мода'**)**

plt**.**legend**()**

AM **=** 0

**for** i **in** range**(**len**(**bins**)):**

**if** bins**[**i**]** **>** M**:**

AM **=** n**[**i**-**1**]**

**break**

IN **=** **(**AM**/**num\_obs**)\***100**/(**2**\***M**/**1000**\*(**dRR**/**1000**))**

plt**.**title**(**'Индекс напряженности равен {}'**.**format**(**IN**))**

plt**.**savefig**(**'INdex.png'**)**

Рисунок 12. Вычисление индекса напряженности по Баевскому

Вывод

В результате разработки VR-игры, использующей биометрические данные, было разработано приложение, применяющее для своего функционирования данные пульса. Был реализован функционал, обеспечивающий регистрацию ЧСС человека при помощи любого устройства, работающего по протоколу Bluetooth Low Energy с использованием спецификации GATT.

Применение данного подхода для опосредованного управления параметрами игры при помощи биометрии человека открывает широкие возможности для применения этой технологии в игровой сфере с целью повышения вовлечения пользователя в процесс или для отслеживания состояния здоровья игрока.

Список использованной литературы

1. <http://protein.bio.msu.ru/~akula/varCI/VarCI.htm> - А.П. Кулаичев «Анализ вариабельности сердечного ритма»
2. <https://docs.microsoft.com/ru-ru/dotnet/csharp/> - Документация по С#
3. <https://docs.unity3d.com/ru/530/Manual/> - Документация по Unity
4. <https://github.com/microsoft/Windows-universal-samples/tree/master/Samples/BluetoothLE/cs> - «Windows-universal-samples»
5. <https://www.vive.com/ru/support/vive/>