**Проект по Компјутерска Графика, летен семестар 2022/23**

**КВИДИЧ ТЕРЕН (QUIDDITCH FIELD)**

**ВО OPENGL**

Ментори:

Лошковска Сузана

Јоксимовски Бобан

**Изработиле:**

**Теодора Ковачка 211172**

**Филип Пенчиќ 211260**

Содржина

[**Краток Вовед** 3](#_Toc156749290)

[**Текот на креирање на проектот** 4](#_Toc156749291)

[**Цртање на теренот со помош на Blender** 5](#_Toc156749292)

[Готовиот 3Д модел за теренот 6](#_Toc156749293)

[**Објаснување за поважните функции и класи** 7](#_Toc156749294)

[Читање на 3Д објектите: класата Object 7](#_Toc156749295)

[Вовед во класата Object 7](#_Toc156749296)

[Функцијата Object::load\_model 8](#_Toc156749297)

[Функцијата Object::set\_buffer\_data 8](#_Toc156749298)

[Интерактивно движење низ сцената: класа Camera 9](#_Toc156749299)

[Функцијата Camera::ProcessKeyboard 9](#_Toc156749300)

[Функцијата ToggleFlightMode 10](#_Toc156749301)

[**Финалниот продукт** 11](#_Toc156749302)

[**Референци** 12](#_Toc156749303)

# **Краток Вовед**

Идејата за проектот потекнува од франчизата Хари Потер (анг. Harry Potter), поточно од измислениот спорт наречен Квидич (анг. Quidditch). Правилата на спортот се едноставни: играчот треба да даде гол во еден од трите обрачи на спротивниот тим, додека лета на метла. Кратко видео од спортот може да погледнете на следниот [линк](https://www.youtube.com/watch?v=fP5UM4elIes&ab_channel=4KHDRMedia).

Целата околина за Квидич спортот е соствена од: терен каде што играчите летаат на метли, високи обрачи (голови), а околу целиот терен се поставени високи кули каде што седи публиката (Слика 1).

Во овој проект нашата цел беше да направиме програма што ќе овозможи движење низ една едноставна верзија од теренот, каде гледачот ќе може да го истражува теренот „пешки“ или „на метла“.



Слика 1: Изглед на теренот за Квидич спортот

# **Текот на креирање на проектот**

Првичниот проект го започнавме со клонирање на проектот кој го користевме на [аудиториските вежби](https://github.com/joksim/OpenGLPrj), специфично на гранката „camera-4“. Во рамките на проектот во папката „vendor“ ја додадовме assimp библиотеката, а во датотеката CMakeLists ги додадовме потребните зависности за истата да биде точно интегрирана во проектот.

Откако успешно ја поврзавме библиотеката, започнавме со имплементирање на читањето на објектите. За таа цел креиравме нови помошни датотеки – Object.hpp и Object.cpp, во чии рамки се имплементира читањето на координатите за позиција на објектите како и нивните бои и текстури и нивно поврзување во OpenGL проектот.

По успешно тестирање на вчитувањето со едноставни 3Д објекти, започнавме со цртање на самиот терен во програмата Blender. Користените слики за текстдодадовме и во рамките на самиот проект во папката res/textures и успешно ги поврзавме истите во датотеката за материјали („field.mtl“).

Откако целиот терен успешно се рендерираше во OpenGL проектот, последната промена што ја направивме беше во рамките на датотеката Camera.cpp, каде што ги променивме функциите за менување на камерата и погледот, за да го добиеме посакуваниот резултат на движење низ теренот „пешки“ и на „метла“. Логиката на движењето е следна:

* Играчот почнува со движење на земја по теренот. Движењето се одвива во прво лице. Може да се движи со копчињата „A“, „W“, „S“ и „D“ од тастатурата, а насоката на движење се менува со движење на глувчето.
* Доколку го притисне копчето „M“ од тастатурата, на играчот му е овозможено   
  „летање“ низ теренот на замислена метла.
* Движењето на метлата се одвива на истиот начин како и движење на земја, само нема ограничување дека играчот мора да се движи по х-оската.

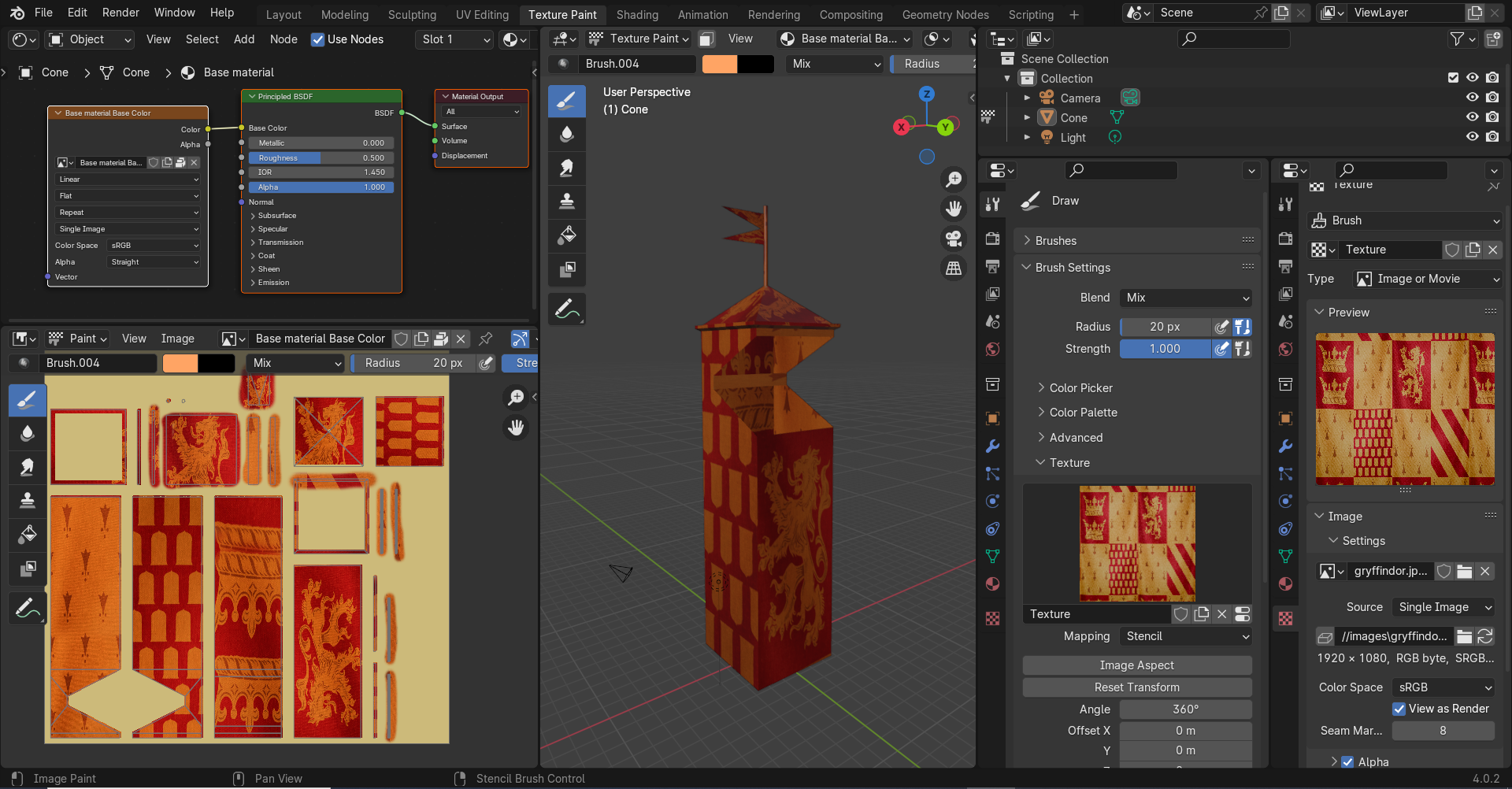
# **Цртање на теренот со помош на Blender**

Со цел да можеме да креираме покомплексен терен, одлучивме дека истиот првично ќе го нацртаме со помош на програмата Блендер (анг. Blender). Започнавме со исцртување на кулите и головите, додавање на текстури на истите и на крај поврзување на сите објекти во една единствена 3Д сцена (Слики 2, 3, 4 и 5). Целиот готов терен го експортиравме во една „.obj“ датотека со име „field.obj“, каде што беа зачувани координатите за позиција како и текстури на теренот. Оваа датотека подоцна ја исчитавме во OpenGl проектот, со помош на класата Object.

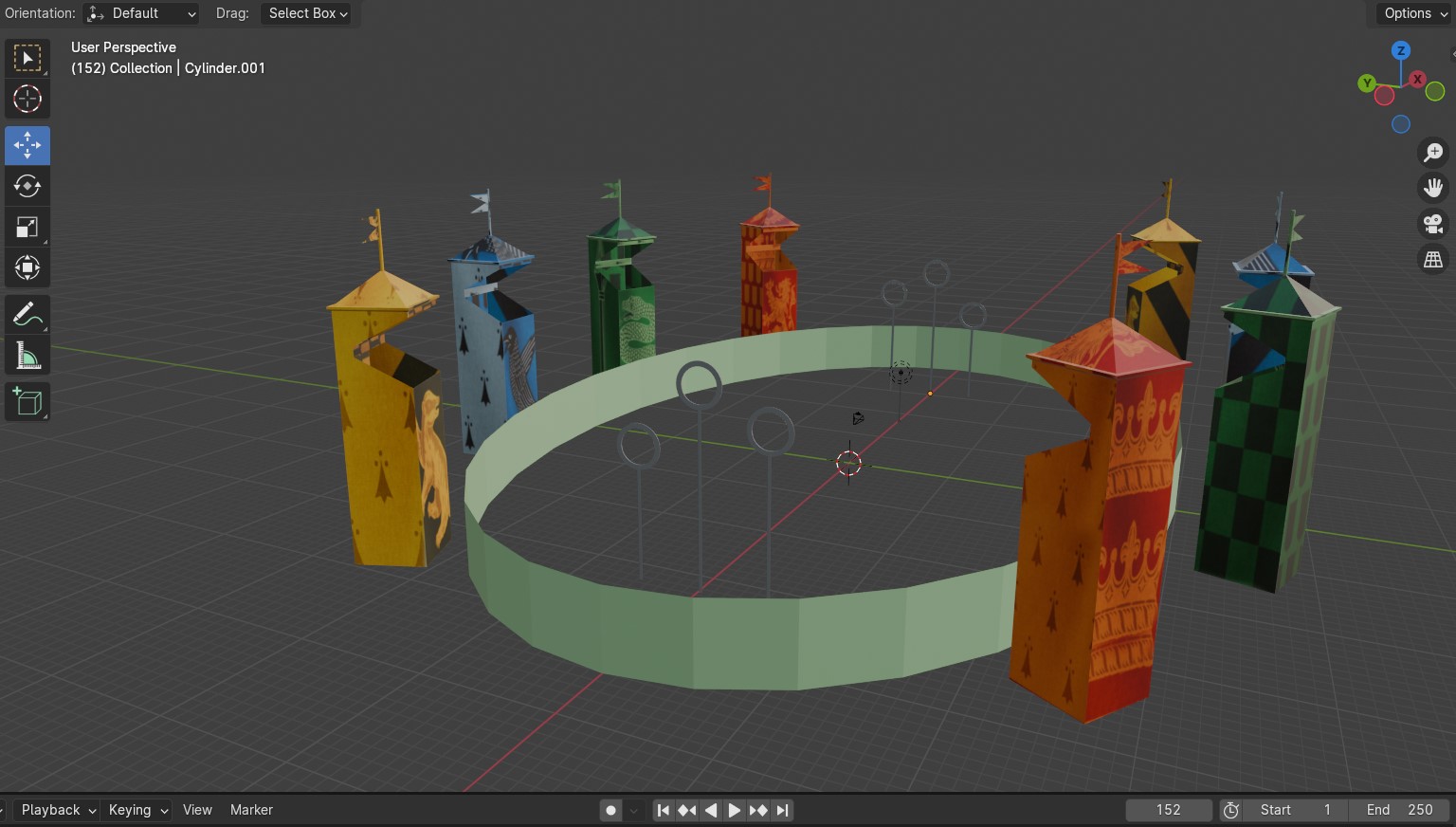
A screenshot of a computer

Description automatically generated

Слика 2: Основно цртање на 3Д формите во Blender



Слика 3: Додавање на текстури за секој 3Д објект во Blender



Слика 4: Склопување на целиот терен во Blender

## Готовиот 3Д модел за теренот

Слика 5 field.obj датотеката прочитана како 3Д објект

# **Објаснување за поважните функции и класи**

## Читање на 3Д објектите: класата Object

За вчитување на целата „field.obj“ датотека во вектори со координати за позиција и текстури, во проектот одлучивме дека ќе ја користиме библиотеката *Open-Asset-Importer-Library -* [*assimp*](https://github.com/assimp/assimp/tree/feb0303fd0ab570ee05c326aa2e39d7312ea932d)*.* За таа цел, креиравме нова класа – *Object* која ќе се справува со читањето од „field.obj“ датотеката и поврзувањето на 3Д објектите во проектот. Декларирањето и имплементацијата на функциите на оваа класа се наоѓаат во датотеките „Object.hpp“ и „Object.cpp“. Во продолжение ќе ја разгледаме оваа класа.

### Вовед во класата Object

Во рамки на *Object* класата, чуваме инстанца од *aiScene* структурата (подоцна ќе ја викаме само *сцена*). *aiScene* е структура во *assimp* библиотеката која нуди олеснување на читањето на датотеки каде се чуваат информации за 3Д објекти, па затоа и ја користиме. Во рамки на конструкторот на *Object,* со помош на класата *Importer* (повторно класа од библиотеката *assimp*), ја читаме нашата датотека „field.obj“ и ја сместуваме во *aiScene* сцената.



Во рамки на *Object* класата чуваме и други помошни структури: *Mesh* и *Texture* за поогранизирано чување на податоците за објектите. Дополнително, чуваме и вектори од сите објекти и сите текстури на сцената, во класата тие се именувани *mesh\_list и texture\_list*.





### Функцијата Object::load\_model

Функцијата *load\_model* која ја повикуваме во конструкторот*,* итеративно изминува низ податоците од *aiScene* сцената и го сместува секој посебен 3Д објект во посебна структура *Mesh* а неговите текстури во структура *Texture*, а сите објекти со нивните текстури ги додава во векторите *mesh\_list* и *tex\_coords* (вектор за координати на текстури на секој објект посебно, деклариран во рамки на *Mesh* структурата). Во следниот дел од кодот се додаваат позициите и текстурите[[1]](#footnote-1) на објектите во рамки на посебни *Mesh* структури, како и во векторите со сите објекти.



### Функцијата Object::set\_buffer\_data

Во рамки на оваа функција ги поврзуваме сите позиции и координати со OpenGl проектот, користејќи ги функциите *glBindBuffer, glBufferData, glEnableVertexAttribArray и glVertexAttribPointer*. Пример за вчитување на позициите на објектите е во следниот код. Слично правиме и за останатите податоци.



## Интерактивно движење низ сцената: класа Camera

Движењето низ сцената беше веќе овозможено во гранката „camera-4“ која ја клониравме од проектот од аудиториските вежби. Првата промена што ја направивме беше во готовата класа *Camera* беше додавање на информација дали играчот се движи пешки (*HUMAN\_WALKING*) или лета (*FLYING*) за полесно справување со двете ситуации.



### Функцијата Camera::ProcessKeyboard

Во рамки на оваа функција, во зависност од вредноста на атрибутот *mode,* движењето преку тастатура или е ограничено на движење само по x-оската, или е целосно слободно за движење во сите правци (вака веќе и беше имплементирано). Во продолжение е кодот за ограничување на движењето во случајот *HUMAN\_WALKING.*



### Функцијата ToggleFlightMode

Во оваа функција е овозможено менување на начините на движење наизменично при секое повикување на функцијата, т.е. доколку начинот бил *HUMAN\_WALKING* по повикот на функцијата ќе биде *FLYING* и обратно.

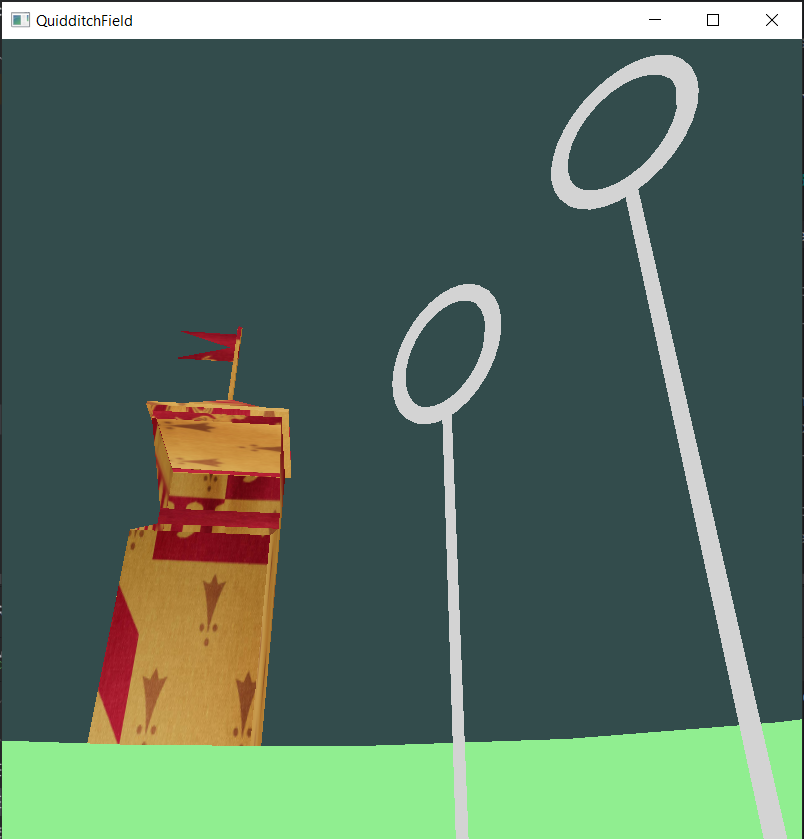


Дополнително, оваа функција ја повикуваме на секое притискање на копчето „М“ од тастатурата, т.е. во функцијата *processInput()* во *main* функцијата.

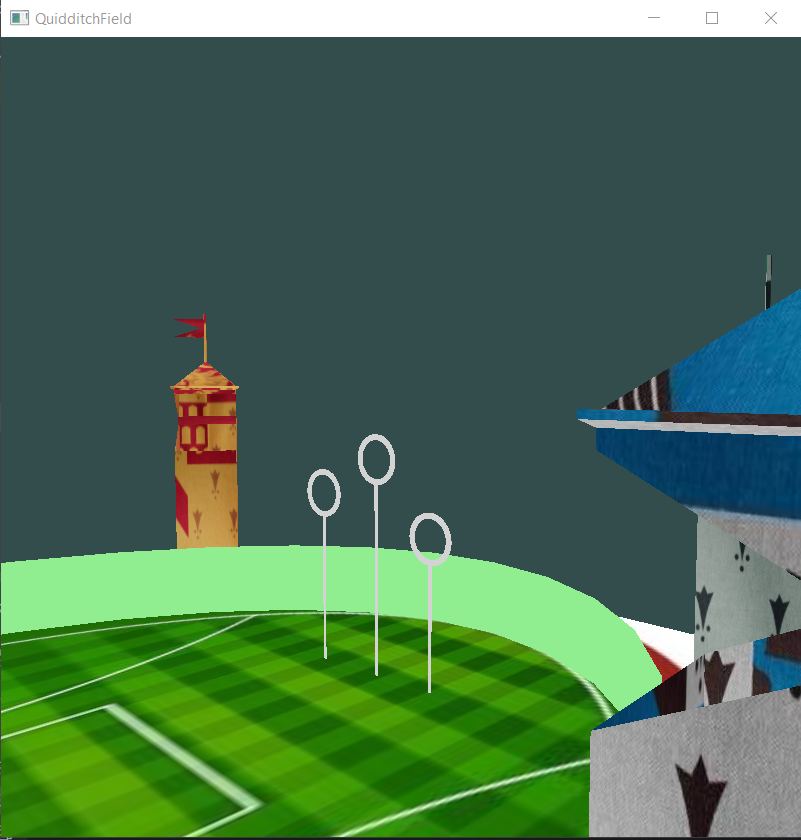


# **Финалниот продукт**

Финалниот продукт што се добива откако ќе се изврши програмата е целиот Квидич терен, рендериран со помош на OpenGl, каде што играчот може да се движи низ истиот и да го истражува пешки (Слика 6) или со „летање“ (Слика 7).



Слика 6 Поглед нагоре, од „пешачење“



Слика 7 Поглед надолу, од „летање“

# **Референци**

1. За почетниот проект

<https://github.com/joksim/OpenGLPrj>

1. За поврзување на assimp библиотеката и менување на CMakeLists

<https://github.com/Polytonic/Glitter>

1. За креирање на Object.hpp и Object.cpp класите

<https://www.programmingcreatively.com/opengl-tutorial-5-qs.php>

<https://www.youtube.com/watch?v=GovbphOagoQ&t=604s&ab_channel=CodeImposter>

1. За цртање на теренот во Blender

<https://www.youtube.com/watch?v=YoikJKRCq34&t=360s>

1. За додавање текстури во Blender

<https://www.youtube.com/watch?v=lO8qDNZf5o0&t=804s>

1. За менувањето на погледот и камерата

<https://stackoverflow.com/questions/71707566/opengl-first-person-realistic-keyboard-movement>

1. Забелешка: се додаваат и податоци за нормалите на објектите, но бидејќи ние во овој проект не работиме со светлина, во овој случај се игнорираат. Секако, доколку се импортира друга датотека што има нормали, истите би биле успешно прочитани. [↑](#footnote-ref-1)