Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра математической
кибернетики и компьютерных наук

РАЗРАБОТКА ИГРОВОГО ДВИЖКА

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Герасимова Владислава Сергеевича

Научный руководитель
зав. кафедрой, к. ф.-м. н. _____ С. В. Миронов
05.06.2017

Заведующий кафедрой
к. ф.-м. н. _____ С. В. Миронов
05.06.2017

Саратов 2017

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

При разработке программного продукта всегда были важны уменьшение стоимости и упрощение самого процесса разработки. В настоящее время существует большое количество решений для достижения этих целей, в том числе и программных. В разработке игр главным таким решением является использование специальных игровых движков.

Целью данной работы является исследование архитектуры различных игровых движков. На основе данного исследования необходимо разработать собственную архитектуру, которая позволит реализовать приложение с минимальным функционалом игрового движка.

Целью практической части работы является разработка собственного игрового движка.

В рамках посталенных целей были определены следующие задачи:

— разработать архитектуру движка;
— реализовать интерфейсы игровых объектов;
— реализовать реализовать механизмы отображения игровых объектов;
— реализовать подсистему загрузки игровой сцены из файлов;
— реализовать возможность управления объектами при помощи сценариев, написанных на скриптовых языках программирования.

1 Игровой движок

В первую очередь, для введения в контекст работы, в данной главе дано определение такого программного продукта, как игровой движок. Описана его основная функциональность, а также подробно рассмотрены различные компоненты игрового движка и их роль в готовой системе.

Далее рассмотрены несколько популярных игровых движков, описаны их основные особенности и отличия [1, 2].

В заключении данной главы рассказано о том, почему игровые движки используются, и приведены основные преимущества их использования.

2 Архитектура разрабатываемого игрового движка

В данной главе описываются основные архитектурные особенности разрабатываемого игрового движка.

Необходимо отметить, что в данной работе рассматриваются такие компоненты игрового движка как графический 3D движок и модуль обработки сценариев, так как они являются основными и с их помощью можно создавать вполне функционирующие игры. Так, например, физическое взаимодействие объектов можно симулировать при помощи сценариев.

В первую очередь в данной главе определяются компоненты, из которых должна состоять игра. Формально игра состоит из набора сцен, сменяющих друг друга в определенные моменты времени. Таким образом, первый компонент, представляющий всю игру в целом, это **Game** (рис. 1).

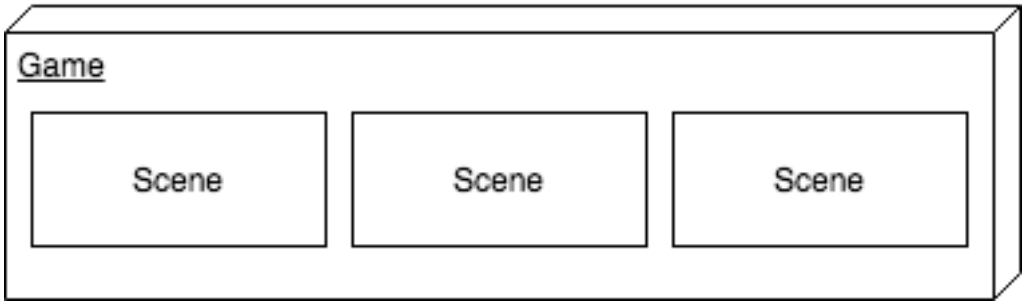

Рисунок 1 – Компоненты игры

Далее определяется, что из себя может представлять каждая из сцен. Подробно описываются составляющие компоненты сцены, которые показаны в виде некоторой удобной иерархии, представленной на упрощенной UML диаграмме, изображенной на рисунке 2.

В заключении данной главы приводится окончательный список компонентов, из которых в разрабатываемом движке состоит игра.

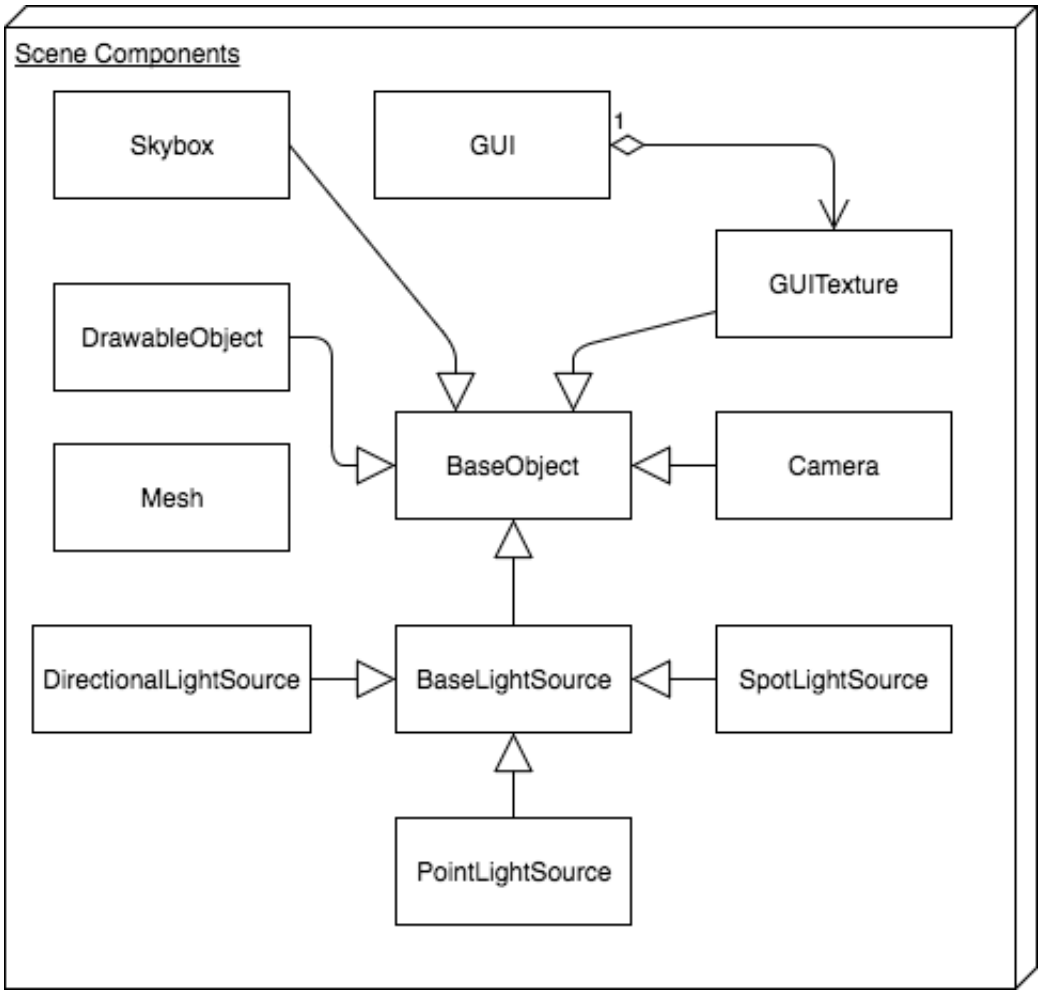

Рисунок 2 – Базовые компоненты игровой сцены

3 Графическая подсистема игрового движка

Одной из центральных тем данной работы является разработка графической подсистемы игрового движка.

В первую очередь в данной главе приводится определение графическому движку. Приводятся отличительные особенности <игровых» графических движков от неигровых [3].

Далее приводится несколько примеров готовых графических движков, которые можно использовать в собственных разработках, описываются некоторые их особенности. Однако они в зачастую являются довольно массивными и сложными продуктами, поэтому было в данной разрабоке лучше написать свое собственное решение, используя возможности графической библиотеки OpenGL.

Далее приводится определение OpenGL. Объясняются главные принципы работы данной библиотеки. Приводятся два основных пути работы с OpenGL: при помощи фиксированного или программируемого конвейера [4].

Описывается архитектура как фиксированного, так и программируемого конвейеров, несколько упрощенный вариант которой представлен на рисунке 3.

Далее описывается, собственно, процесс построения изображения конвейером OpenGL.

В заключении данной главы приводятся отличия программируемого конвейера от фиксированного, а также некоторые особенности работы с программируемым конейером со стороны разработчика приложения.

3.1 Архитектура графического движка

В данной главе описываются архитектурные особенности конкретно графической подсистемы разрабатываемого игрового движка.

Все отображаемые объекты так или иначе состоят из какого-то количества вершин, которые при рендеринге будут объединяться в полигоны (обычно в треугольники). Поэтому в первую очередь определяется то, как объекты будут храниться в памяти движка.

Далее приводятся несколько особенностей работы по пересылке отображаемых данных в видеопамять. Подробно описываются такие объекты как Vertex Array Object (VAO) и Vertex Buffer Object (VBO).

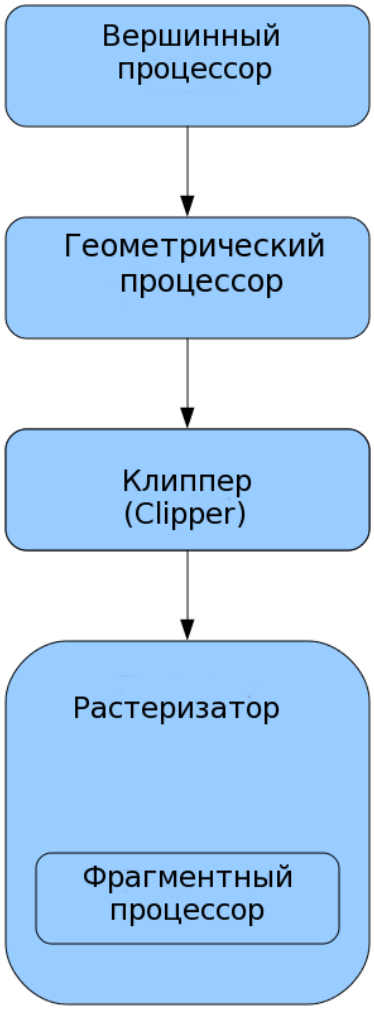

Рисунок 3 – Конвейер OpenGL

Далее определяется алгоритм по работе с шейдерами в приложении и приводится определение шейдера.

3.2 Различные техники, использующиеся в графических движках

В данной главе описываются некоторые техники и особенности по построению изображений в графический движках.

3.2.1 Модели освещения

На данный момент существует большое количество моделей освещения различной сложности. В первую очереь приводятся некоторые примеры моделей освещения.

Одной из самых распространенных моделей освещения является модель Фонга. В данной работе используется именно она [5–7]. Модель освеще-

ния Фонга требует сравнительно немного ресурсов компьютера, однако большинство оптических явлений игнорируются либо рассчитываются с грубым приближением.

Далее описываются составляющие итоговой освещенности объектов в модели Фонга — фоновая (ambient), рассеянная (diffuse) и глянцевая (specular).

Далее приводится формула 1 рассчета освещенности точки на поверхности какого-то объекта.

$$I = K_a I_a + K_d(\vec{n}, \vec{l}) + K_s(\vec{n}, \vec{h})^p, \tag{1}$$

где

$\vec{n}$ — вектор нормали к поверхности в точке;

$\vec{l}$ — направление проецирования (направление на источник света);

$\vec{h}$ — направление на наблюдателя;

K_a — коэффициент фонового освещения;

K_s — коэффициент зеркального освещения;

K_d — коэффициент диффузного освещения.

3.2.2 Источники света

Формула 1 показывает общий случай освещения любыми источниками света. В данной главе описываются различные типы источников света [8]:

— направленный источник света (Directional light source);

— точечный источник света (Point light source);

— прожектор (Spotlight source).

3.2.3 Рельефное текстурирование

В компьютерной графике часто применяются различные методы рельефного текстурирования для придания большей реалистичности объектам за счет добавления дополнительной информации о поверхности объекта.

В данной главе приводится определение и возможности такой техники рельефного тектурирования, как normal mapping [9–11].

3.2.4 Гамма-коррекция

Глаза воспринимают света не так, чем камеры. В цифровой камере удвоенное количество фотонов, попадающих на сенсор, означает удвоение сигнала — линейная зависимость. Однако глаза устроены иначе. Увеличение осве-

щённости вдвое означает, что свет стал лишь слегка ярче. В данной главе приводится решение данной проблемы при помощи такого искажения яркости чёрно-белого или цветоделённых составляющих цветного изображения как гамма-коррекция [12].

Приводится формула 2, определяющая степень коррекции изображения.

$$V_{\text{out}} = AV_{\text{in}}{}^{\gamma}, \tag{2}$$

где A служит коэффициентом, а входные V_{in} и выходные V_{out} значения — неотрицательные вещественные числа.

3.2.5 Карта теней

Само по себе освещение объектов в компьютерной графике не подразумевает отбрасывание ими теней. Однако тени необходимы для придания изображению большей реалистичности. В данной главе описывается такой метод создания теней как shadow mapping (карта теней).

Подробно рассматривается процесс построения карты теней для одного источника света [13–15]. Кроме того, отмечается разница в построении карт теней для различных типов источников освещения.

4 Разработка игрового движка

Данная глава является описанием практической части настоящей работы, то есть напосредственно самой разработки игрового движка.

4.1 Программное окружение разрабатываемого движка

В данной главе приводится описание программное окружение разрабатываемого приложения. В частности оисываются операционная система, в которой ведется разработка, язык разработки, версии компилятора и системы сборки и пр.

Кроме того перечисляются дополнительных библиотеки, используемые в разработке, а также описывается их функциональность [16, 17, 17–32].

4.2 Полезные вспомогательные классы

В разработке приложения будет полезно иметь некоторые вспомогательные классы, упрощающие написание остального кода. В данной главе описываются такие вспомогательные классы.

В разрабатываемом движке типом хранения игры в постоянной памяти является XML. Поэтому, кроме всех прочих, водится и подробно описывается класс `XMLReader` для работы с документами такого типа.

4.3 Разработка графического движка

В данной главе описываются основные классы, составляющие графическую подсистему разрабатываемого игрового движка.

В первую очередь, это классы-обертки для удобной работы с шейдерами `Shader` и `ShaderProgram`. Приводится их подробное описание, а также важные моменты в реализации их методов.

Следующим этапом приводится разработка самих шейдеров.

В первую очередь определяется, какие шейдеры необходимы и для чего.

— `simple` — отвечает за непосредственно отрисовку всех объектов;
— `shadow` — отвеччает за отрисовку объектов в карты теней;
— `skybox` — отвечает за отрисовку скайбокса;
— `gui` — отвечает за отрисовку графического интерфейса.

Далее описывается непосредственно разработка каждого из вышеобозначенных шейдеров.

Следующим этапом описывается пример использования шейдеров при выводе изображения объектов на экран. Приводятся важные моменты в реализации механизмов отображения объектов.

Далее вводятся и подробно описываются дополнительные классы для работы со списком шейдеров, для чтения конфигураций шейдеров из XML-файлов, а также для работы с окном, в котором происходит отображение игры.

4.4 Промежуточный результат

В данной главе подводятся промежуточные итоги о реализованному к данному моменту функционалу игрового движка.

Также приводится пример использования полученной системы.

4.5 Игровые объекты

Следующим шагом будет разработка собственно игровых объектов (камеры, источники света и др.), сцены и самой игры, согласно архитектуре, принятой в разделе 2.

В первую очередь в данной главе определяется то, как будут производиться следущие действия:

1. отображение сцен на экране;
2. обработка сценариев.

4.5.1 Отрисовка объектов

В данной главе, учитывая архитектуру, разработанную в разделе 2, вводится и подробно описывается последовательность действий для отображения сцен и их объектов.

4.5.2 Обработка сценариев игры

В данной главе, в свою очередь, определяется и описывается последовательность действий для обработки сценариев игры.

4.5.3 Разработка компонентов игры

Далее можно переходить к разработке основных компонентов игры. Предназначение каждого из них было описано в разделе 2, поэтому в данном разделе приводятся основные наборы их полей и методов.

Важным этапом является работы движка является создание всех игровых объектов согласно конфигурации, заданной во входном XML-файле. Для этого вводится и описывается класс `GameReader`. Его основной целью является чтение всего файла и добавление прочитанных объектов в глобальный объект Game.

4.6 Модуль обработки сценариев

Следующим этам является разработка функционала модуля обработки сценариев.

В качестве скриптового языка программирования выбран ChaiScript [22] как наиболее удобный при привязывании функций и классов C++.

В данной главе приводится описание работы с ChaiScript при реализации функционала по обработки игровых сценариев.

5 Демонстрация работы с разработанным движком

В качестве демонстрации работы разработанного движка в данной шлаве с его помощью приводится пример создания небольшой простой игры.

Приводится подробное описание XML-файла с конфигурацией игры, а также файла с основным сценарием игры.

В заключении приводятся скриншоты, демонстрирующие работоспособность игры, созданной при помощи созданного движка.

ЗАКЛЮЧЕНИЕ

В результате данной работы:

— были изучены различные техники, использующиеся в различных графических движках;
— был разработан собственный графический движок с использованием библиотеки OpenGL;
— была изучена архитектура некоторых популярных игровых движков;
— была разработана собственная архитектура приложения, реализующего минимальный функционал игровых движков;
— на основе полученной архитектуры был разработан собственный игровой движок;
— был существенно улучшен навык разработки на языке C++;
— для демонстрации возможностей полученного приложения с его помощью была построена небольшая простая игра.

На основе полученных результатов можно сделать следующие выводы.

В данной разработке рассматривались только самые главные компоненты игрового движка: графическая подсистема и модуль обработки сценариев. В дальнейшем возможно добавление других компонентов, например, интересной задачей будет разработка физического движка, либо же использование готового.

В настоящее время набирает популярность низкоуровневая альтернатива OpenGL — Vulkan. Для обеспечения большей производительности и кроссплатформенности можно реализовать графический движок, используя именно Vulkan или, например, DirectX.

Игровой движок является большим и комплексным программным продуктом. Несмотря на это, задача по разработке игрового движка вполне может быть выполнена одним человеком. Благодаря тому, что она содержит довольно много различных аспектов создания приложений, разработка игрового движка может выступать в качестве учебного проекта, а данная работа может быть использована в качестве методического пособия.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 23 Recommended 3D Game Engines (Updated) [Электронный ресурс]. — URL: http://www.worldofleveldesign.com/categories/level_design_tutorials/recommended-game-engines.php (Дата обращения 20.04.2017). Загл. с экр. Яз. англ.

2 Engines for Games - Indie DB [Электронный ресурс]. — URL: http://www.indiedb.com/engines (Дата обращения 20.04.2017). Загл. с экр. Яз. англ.

3 Графические движки [Электронный ресурс]. — URL: http://bourabai.ru/graphics/graphengine.htm (Дата обращения 12.04.2017). Загл. с экр. Яз. англ.

4 Shader - OpenGL Wiki [Электронный ресурс]. — URL: https://www.khronos.org/opengl/wiki/Shader (Дата обращения 12.04.2017). Загл. с экр. Яз. англ.

5 Модель отражения Фонга :: Освещение :: Теория 3D :: Компьютерная графика [Электронный ресурс]. — URL: http://compgraphics.info/3D/lighting/phong_reflection_model.php (Дата обращения 15.01.2017). Загл. с экр. Яз. рус.

6 The Phong Model, Introduction to the Concepts of Shader, Reflection Models and BRDF (The Phong Model and the concepts of Illumination Models and BRDF) [Электронный ресурс]. — URL: https://www.scratchapixel.com/lessons/3d-basic-rendering/phong-shader-BRDF (Дата обращения 15.01.2017). Загл. с экр. Яз. англ.

7 Phong Model - Specular Reflection [Электронный ресурс]. — URL: https://www.siggraph.org/education/materials/HyperGraph/illumin/specular_highlights/phong_model_specular_reflection.htm (Дата обращения 15.01.2017). Загл. с экр. Яз. англ.

8 Электронный учебник «Компьютерная графика» [Электронный ресурс]. — URL: goo.gl/oFp0B8 (Дата обращения 02.02.2017). Загл. с экр. Яз. рус.

9 Normal Mapping [Электронный ресурс]. — URL: https://learnopengl.com/#!Advanced-Lighting/Normal-Mapping (Дата обращения 25.04.2017). Загл. с экр. Яз. англ.

10 Gamedev Glossary: What Is a “Normal Map”? [Электронный ресурс]. — URL: https://gamedevelopment.tutsplus.com/articles/gamedev-glossary-what-is-a-normal-map--gamedev-3893 (Дата обращения 25.04.2017). Загл. с экр. Яз. англ.

11 Difference between Displacement, Bump and Normal Maps | Pluralsight [Электронный ресурс]. — URL: https://www.pluralsight.com/blog/film-games/bump-normal-and-displacement-maps?utm_source=rss&utm_medium=rss&utm_campaign=bump-normal-and-displacement-maps (Дата обращения 25.04.2017). Загл. с экр. Яз. англ.

12 Что такое гамма-коррекция [Электронный ресурс]. — URL: http://www.cambridgeincolour.com/ru/tutorials-ru/gamma-correction.htm (Дата обращения 04.02.2017). Загл. с экр. Яз. рус.

13 Shadow Mapping [Электронный ресурс]. — URL: https://learnopengl.com/#!Advanced-Lighting/Shadows/Shadow-Mapping (Дата обращения 25.04.2017). Загл. с экр. Яз. англ.

14 Coding Labs :: Deferred Rendering Shadow Mapping [Электронный ресурс]. — URL: http://www.codinglabs.net/tutorial_opengl_deferred_rendering_shadow_mapping.aspx (Дата обращения 25.04.2017). Загл. с экр. Яз. англ.

15 OpenGL Shadow Mapping Tutorial - Paul’s Projects [Электронный ресурс]. — URL: http://www.paulsprojects.net/tutorials/smt/smt.html (Дата обращения 25.04.2017). Загл. с экр. Яз. англ.

16 GLEW: The OpenGL Extension Wrangler Library [Электронный ресурс]. — URL: http://glew.sourceforge.net/ (Дата обращения 10.01.2017). Загл. с экр. Яз. англ.

17 nigels-com/glew: The OpenGL Extension Wrangler Library [Электронный ресурс]. — URL: https://github.com/nigels-com/glew (Дата обращения 10.01.2017). Загл. с экр. Яз. англ.

18 GLFW - An OpenGL library [Электронный ресурс]. — URL: http://www.glfw.org/ (Дата обращения 10.01.2017). Загл. с экр. Яз. англ.

19 hbristow/argparse: A slimline C++ class for parsing command-line arguments, with an interface similar to python’s class of the same name

[Электронный ресурс]. — URL: https://github.com/hbristow/argparse (Дата обращения 10.01.2017). Загл. с экр. Яз. англ.

20 assimp/assimp: Official Open Asset Import Library Repository. [Электронный ресурс]. — URL: https://github.com/assimp/assimp (Дата обращения 10.01.2017). Загл. с экр. Яз. англ.

21 Easylogging++ - Single header only, cross-platform logging library for C++ applications [Электронный ресурс]. — URL: https://muflihun.github.io/easyloggingpp/ (Дата обращения 10.01.2017). Загл. с экр. Яз. англ.

22 ChaiScript [Электронный ресурс]. — URL: https://github.com/ChaiScript (Дата обращения 10.01.2017). Загл. с экр. Яз. англ.

23 ChaiScript - Easy to use scripting for C++. [Электронный ресурс]. — URL: http://chaiscript.com/ (Дата обращения 10.01.2017). Загл. с экр. Яз. англ.

24 C++ embedded scripting: ChaiScript recipes | Aleksey Fedotov [Электронный ресурс]. — URL: http://f5v.me/chaiscript-recipes (Дата обращения 10.01.2017). Загл. с экр. Яз. англ.

25 OpenGL Mathematics [Электронный ресурс]. — URL: http://glm.g-truc.net/0.9.8/index.html (Дата обращения 10.01.2017). Загл. с экр. Яз. англ.

26 GLM SDK contribution [Электронный ресурс]. — URL: https://www.opengl.org/sdk/libs/GLM/ (Дата обращения 10.01.2017). Загл. с экр. Яз. англ.

27 g-truc/glm: OpenGL Mathematics (GLM) [Электронный ресурс]. — URL: https://github.com/g-truc/glm (Дата обращения 10.01.2017). Загл. с экр. Яз. англ.

28 Boost C++ Libraries [Электронный ресурс]. — URL: http://www.boost.org/ (Дата обращения 10.01.2017). Загл. с экр. Яз. англ.

29 The Boost C++ Libraries [Электронный ресурс]. — URL: https://theboostcpplibraries.com/ (Дата обращения 10.01.2017). Загл. с экр. Яз. англ.

30 Boost.org [Электронный ресурс]. — URL: https://github.com/boostorg (Дата обращения 10.01.2017). Загл. с экр. Яз. англ.

31 Magick++ API [Электронный ресурс]. — URL: http://www.imagemagick.org/Magick++/ (Дата обращения 10.01.2017). Загл. с экр. Яз. англ.

32 TinyXML-2 [Электронный ресурс]. — URL: http://www.grinninglizard.com/tinyxml2/ (Дата обращения 10.01.2017). Загл. с экр. Яз. англ.

05.06.2017