1. Введение:

Целью создания данной спецификации является описание архитектуры и функционала компьютерной игры "Сапёр 3D", использующей высокоуровненые библиотеки С++ для реализации 3D графики.

1. Описание системы:

Данный проект был создан с целью усложнения стандартной версии игры "Сапёр" путём добавления третьего измерения к игровому полю. Из этой особенности вытекает необходимость использования средств трёхмерной графики, а также организации игрового поля в понятном игроку виде. Также для облегчения работы с приложением планируется добавление простейшей системы меню и начального экрана.

1. Особенности дизайна:

* Зависимости  
   Разработка приложения будет вестись на языке C++, в среде Visual Studio. Для эффективной реализации 3D рендеринга была выбрана высокоуровневая библиотека Magnum, в своей основе использующая OpenGL. Библиотека является кроссплатформенной и также содержит в себе средства организации игровой сцены, и инструменты взаимодействия с периферией ПК.
* Ограничения  
   Единственным явным ограничением является наличие в системе графического ускорителя с поддержкой стандарта OpenGL, т.к. библиотека Magnum является кроссплатформенной.
* Цели и рекомендации

Игровое поле должно быть представлено в доступной игроку форме, предоставляя максимум информации и при этом оставаясь визуально простым;

Система меню должна быть предельно простой и предоставлять базовые действия - начало игры, настройку сложности, перезапуск игры и выход из приложения.

1. Архитектурные решения:

Функционал приложения будет разделён по сценам - отдельным объектам, содержащим в себе все необходимые ресурсы для представления очередной стадии работы приложения;

В качестве основы пользовательского интерфейса будет использоваться модуль ImGUI вместо стандартного модуля Magnum, находящегося ещё в незавершенном состоянии;

Для простоты организации игрового поля, его составляющие будут объединены в граф сцены - структуру, упрощающую работу с группами объектов, а также с объектом камеры.

1. Системная архитектура:

* Организация меню:

Система меню будет представлена главным меню, выделенным в отдельную сцену, и простейшими окнами появляющимися во время игры - меню паузы и меню победы/поражения.

* Организация графа сцены:

Граф сцены в общем смысле представлен одним корнем и множеством произвольно соединенных объектов. У каждого объекта может быть лишь один предок и произвольное число потомков. Каждый объект хранит внутри себя полную ориентацию относительно своего предка, что значительно упрощает управление группой связаных объектов и нахождение относительной ориентации между любыми объектами графа.

В данном приложении граф будет служить для организации кубов-клеток поля в упорядоченный массив. С целью лучшей их организации будут введены вспомогаельные объекты для каждого горизонтального слоя.

* Инициализация игрового поля:

После перехода на сцену игрового поля и построения сетки кубов-клеток, необходимо заполнить их минами. Из-за добавления третьего измерения, даже небольшое поле будет содержать значительное число клеток. Для уменьшения шанса получения очень разреженого поля было решено располагать часть мин в более тесных группах, и разбрасывать остальные по случайным клеткам.

1. Подробное описание дизайна:

* Организация графа сцены

В Magnum, весь функционал связанный с графом сцены размещён в пространстве Magnum::SceneGraph. Его можно разделить на 3 группы:

* Объекты, предоставляющие иерархию предок-потомок

Они представлены двумя классами - Scene и Object, являющиеся соответственно корнем графа и минимальным вариантом объекта графа. При создании им нужно предоставить тип трансформации, которые будут рассморены ниже. При создании графа можно хранить в переменной лишь его корень - каждый объект хранит список своих потомков, что позволяет находить любой объект графа.

* Трансформации, определяющие конкретную реализацию ориентации объектов графа

Они используются как параметр шаблона при создании объекта. Трансформации задают размерность графа - 2D или 3D, и имеют свои достоинства и недостатки. Например, MatrixTransformation3D поддерживает произвольные трансформации, но затратна при вычислении обратной трансформации. В то же время DualQuaternionTransformation поддерживает лишь смещение и вращение, но быстро проводит операцию обращения.

* Доп. функционал, добавляющий широкий набор возможностей к базовому объекту графа

Он представлен набором специализированных классов, каждый добавляющий определенные свойства к объекту Object. Например, класс Drawable используется для отрисовки объектов на экране, класс Camera служит окном через которое возможна отрисовка графа, класс Animable добавляет возможность анимациии.

Добавление функционала к объекту возможно двумя способами - созданием класса-наследника Object, наследующего также весь функционал, и присоединением объекта нужной функции к базовому объекту во время выполнения программы.

В контексте данного приложения граф сцены будет использоваться для организации кубов-клеток поля сначала в горизональные слои, затем в вертикальную стопку. Это позволит легче организовать управление камерой и отображение отдельных слоёв.

* Инициализация игрового поля:

Блок-схема алгоритма генерации кластера приведена в приложенном файле. Он запускается в ходе подготовки игрового поля со случайными координатами.