Министерство образования и науки Российской Федерации

ГОУ ВПО «Уральский федеральный университет

им. первого Президента России Б.Н. Ельцина»

Кафедра Интеллектуальных информационных технологий

**Summer Game**

Отчет по учебной практике

Руководитель проекта Кругликов Алексей Сергеевич

Студенты Бетин Егор Андреевич

Василенко Михаил Алексеевич

Змеев Виктор Викторович

Харрасов Алмаз Альфисович

Группа ФО-311001

**Введение**

На летнюю практику была поставлена задача реализовать игру, имитирующую события Второй Мировой Войны. Была поставлена задача реализовать логическую модель игры и интерфейс для взаимодействия с пользователем.

Концепция игры заключается в ведении боевых действий на игровом поле, используя пошаговый режим управления боевыми единицами во времена Второй Мировой Войны с учетом всех временных эпизодов с участием соответствующих сторон. В качестве карты была использована схематичная карта Халхин-Гола со всеми происходившими в данной местности событиями. В качестве боевых единиц были выбраны группа Быкова и две пехотные японские роты. По реальным данным были сформированы различные условия местности, оружия, техники, которые отличаются совокупностью свойств и влияний на боевую механику боя.

**Цели:**

1. Сформировать концепцию игры;
2. Разработать архитектуру игровой модели;
3. Реализовать логическую модель для осуществления игровых взаимодействий;
4. Реализовать пользовательский интерфейс;
5. Интегрировать логику игры в пользовательский интерфейс посредством промежуточной системы.

**Архитектура игры SummerGame**

**Абстрактная модель игры**

Решение SummerGame состоит из нескольких проектов:

* AbstractGameEngine;
* GameEngine;
* Geometry;
* UserInterface;
* SummerGame.

Логическую часть решения содержат проекты: AbstractGameEngine, GameEngine, Geometry.

Подробней об AbstractGameEngine. Это проект, который содержит в себе интерфейсы и абстрактные классы для создания конкретных классов. Так как задумка с самого начала рассчитана на игру разновидности «стратегия» в этом проекте присутствуют такие сущности, как *Administrant* и *Character*.

*Administrant* является сущностью исполнителя каких-либо действий, то есть просто умеет совершать различные действия, а также имеет в себе сущность *Side*, которая отвечает за принадлежность его к чему или кому-либо, так как это – абстрактная модель, то мы не можем окончательно сказать по поводу того, будет ли принадлежность, какая она будет или не будет совсем.

*Character* же более конкретное представление участника игры, он имеет набор свойств, который также основан на сущности *UnitProperties*.

Для каждого из этих сущностей должно быть задано правило действия для того, чтобы в игре появились хоть какие-то движения или цели. Для этого введена сущность *Action*, которая явно говорит о своем значении.

*Action* имеет в себе цель исполнять любые запросы персонажей игры, поэтому существует параметризованный вид *Action*, чтобы в него можно было поместить любое действие с помощью делегата. Также *Action* имеет в себе состояние действия, закончилось ли оно, время исполнения, и , конечно, методы для исполнения действия в зависимости от делегата.

Чтобы персонажи могли что-то делать, им нужно где-то это делать, поэтому возникает прямая необходимость ввести сущность *Area*, которая будет просто отвечать за занятый участок карты каким-либо объектом, персонажем и так далее.

*Area* – это сущность занятой области и только в ней использованы сущности проекта Geometry. Причина, по которой в абстрактной модели игры используются конкретные классы проекта Geometry интуитивно понятна. Нам нужны фигуры для описания области.

Конечно, нам в игре нужны предметы, поэтому была введена сущность *Item*. Ничего особенного она собой не представляет. Может лишь занимать какую-либо область.

Далее предполагалось, что персонажи будут общаться между собой, поэтому появилась сущность *Message*, она несет в себе какой-то *Action*, а также информацию о том, кто послал сообщение, и кому.

Так как игра разновидности «стратегия», то нужно иметь ввиду, что нам нужно объединять персонажей в группы, за это будет отвечать *Squad*.

На начальном этапе в абстрактной модели также существовала сущность *Game*, которая в итоге оказалась излишней и была исключена. После же была реализована, как вспомогательный класс в логике игры (проект GameEngine).

**Логика игры**

Игровая логика заключена и конкретизирована в проекте GameEngine, который содержит в себе различные классы, в том числе те, которые способствуют проведению игры, а не участвуют в ней.

В GameEngine описаны классы, реализующие абстрактную модель и различные взаимодействия, а также в ней присутствуют отдельные блоки конкретизации формул, расчетов игровых, и, конечно, алгоритмы и методы для обработки игровых событий.

Логика игры описывается в несколько блоков:

* Реализация персонажей (Character);
* Реализация Action, различных действий доступных персонажу;
* Реализация различных видов местности (Lands от Area);
* Реализация предметов для игры (Object от Item);
* Классы, отвечающие за формирование историй, которые передаются интерфейсу;
* Расширения и статический класс для Загрузки игры и сохранения;
* Класс Game, который отвечает за обсчет игрового времени и высылку историй интерфейсу.

**Подробнее о составляющих GameEngine:**

Реализация персонажей – создание конкретных классов войск для игры SummerGame.

Реализация Action – возможность передвигаться и атаковать.

Реализация местности – ограничение скорости передвижения по разным местностям.

Реализация предметов игры – амуниция для боя, техника и прочее.

Формирование историй – создание набора действий на промежуток времени для игрового интерфейса, чтобы пользователь мог видеть происходящие события игровой модели.

Расширение и класс сериализатор – для загрузки и сохранения объектов карты, персонажей (войск), а также областей карты, местностей.

Game – содержит в себе принадлежность хода игроку, отсчет игрового времени и карту, карта представляет из себя набор персонажей (войск), предметов, местностей. В методе подсчета нового момента игрового времени этот класс отсылает набор историй проекту UserInterface для интерпретации на экране пользователя.

**Geometry**

Проект Geometry занимается вспомогательными расчетами для формирования действий Unit’a. Реализует формирование пути через различные области карты, расчет расстояний пути по конкретной области, точки входа и выхода из области.

Любой элемент карты (*Area*) – композиция примитивных фигур.

Геометрические фигуры, входящие в проект Geometry:

* Точка – обусловлена 2мя координатами;
* Отрезок – обусловлен 2мя точками;
* Круг – обусловлен точкой и радиусом;
* Ломаная – обусловлена списком точек;
* Выпуклый многоугольник – обусловлен списком точек и списком нормалей ребер.

Для основных вычислений используются переопределенные методы *GetIntersection* и *IsIntersected* .

GetIntersection возвращает список всех точек пересечения фигуры1 и фигуры2, а так же все граничные точки, принадлежащие обеим фигурам.

IsIntersected – булевская функция, которая проверяет, пересеклись ли фигуры.

**GameEngine**

*Lands*, или *ландшафт* – это набор различных областей, которые могут находиться на карте. Каждая область обладает своими уникальными параметрами каждого из 3-х основных свойств.

Элементы *Lands* различаются по:

* *Passability*, или *Проходимость*. Отвечает за ограничение максимальной скорости движения Unit`a по конкретной области;
* *Opennes*, или *Открытость*. Отвечает за ограничение максимальной точности стрельбы и усложнение ведения боевых действий в конкретной области;
* *Visibility*, или *Видимость*. Отвечает за ограничение на максимальное расстояние, на котором можно заметить отряд противника.

*Characters*, или *Персонажи* – задает и описывает возможное поведение активных элементов карты. Состоит из:

* *Group* – список группа людей, объединенных по общим параметрам;
* *Features* – набор параметров, описывающий основные свойства боевой единицы;
* *Unit* – минимальная боевая единица, доступная для управления игроком;
* *AttackAction* – занимается расчетом и выполнением боевого взаимодействия различных Unit`ов, принадлежащих к противоборствующим сторонам;
* *MoveAction* – занимается формирование и выполнением движения каждого Unit`a , с учетом формирования пути, изменениями скорости и других факторов.

*Group*, или *группа* – неделимый элемент Unit`a, объединенный по общим признакам.

Group обусловлена:

* *Count* – количество людей в группе;
* *Rank* – текущий ранг людей, состоящих в группе. Различают Ordinary, Sergeant, Officer;
* *Specialization* – специализация людей, состоящих в группе. Группы бывают либо Infantry (пехота), либо Tankman (танкисты);
* *Qualification* – квалификации людей, состоящих в группе. Делится на Low, Medium и High;
* *Experience* – боевой опыт группы. Делится на Low, Medium и High;
* *Vitality* – текущее физическое состояние группы. Влияет на возможность участия группы в сражении. Разделяют на *Healthy*, *wounded*, *HeavilyWounded* и *Dead*.

*Features*, или *свойства*. Основные параметры, которые формируют Unit’a.

Свойства Unit`a:

* *Groups* – список групп, входящих в состав;
* *Items* – список вещей, которыми обладает Unit;
* *CurVisible* – текущая дальность видимости, с учетом различных факторов;
* *CurSpeed* – текущая скорость передвижения, с четом различных факторов.

Unit, как минимальный боевой элемент, доступный для управления игроку, состоит из:

* *Features* – набора свойств;
* *UnitType* – типа боевой единицы;
* *Side* – противоборствующая сторона, к которой принадлежит Unit.

**UserInterface**

Библиотека UserInterface состоит из следующих классов:

1. *UISelection*: целью данного класса является создание списка выделенных фигур (выделение происходит при помощи клика на юнита). Выбранных юнитов можно передвигать по карте или выбирать и атаковать вражеских юнитов;
2. *UI*: при помощи *UI* формируются следующие команды:
   1. Создание истории для движения юнита(юнитов)
   2. Создание истории для атаки юнитом(юнитов) вражеского юнита.

Так же этот класс отвечает за паузу и старт в игре, за предоставление информации при наведении на какую-нибудь местность и за смену стороны, за вызов сохранения и загрузки игры;

1. *Transformer*: Transformer отвечает за конвертацию различных игровых данных, таких как:
   1. Экранные координаты в модельные.
   2. Модельные координаты в экранные.
   3. Преобразование реального времени в игровое.

А также содержит коэффициенты, обозначающие разницу между экранными и модельными координатами;

1. *StoryConverter*: При помощи StoryConverter происходит анимация сформировавшихся историй, таких как атака и передвижение юнитов;
2. Данный модуль содержит конвертеры *AreaToCanvasLeft*, *AreaToCanvasTop*, *CircleToCircle, CircleToX, CircleToY, FigureToCanvasLeft, FigureToCanvasTop, PointToPoint, PolygonToPolygon, PolylineToPolyline, SegmentToSegment*, которые служат для внутренних нужд программы.

**Интерфейс игры**

На данный момент существуют 3 функционирующих окна: *главное меню*, в которое пользователь попадает по запуску игры, *окно игры* – по запуску *Новой игры* и *окно свойств* объекта – по клику ПКМ на объекте.

**Функционал**

Реализован интерфейсный функционал окна главного меню, окна игрового поля, окна свойств выбранного объекта. Стиль для ландшафта задается в зависимости от их типов. Ландшафты, в свою очередь, представляют собой набор фигур, для которых также заданы стили в зависимости от типа фигуры. Каждый тип ландшафта может быть представлен фигурами: polygon, polyline, circle, segment, point. Unit`ы в игре отображаются в зависимости от типа и принадлежности стороне игрока. При наведении курсора мыши на объекты отображаются их типы. При щелчке правой кнопкой мыши на объекте показывается окно с его свойствами.

С помощью ЛКМ делается выбор unit`а на карте. Выбирается место на карте, куда перемещается unit (тоже ЛКМ). Либо выбирается группа из unit`ов и аналогично перемещается. Если же при выборе unit’а нажать ЛКМ на один из unit`ов противника, начинается атака. Пауза в игре устанавливается нажатием клавиши SPACE на клавиатуре, либо в правой колонке по нажатию соответствующей кнопки. В правой колонке можно перейти в главное меню либо перейти в раздел настроек. Также реализовано масштабирование карты, которое также управляется в правой колонке. В верхнем баре отображается краткая информация об объектах, на которые наведен курсор. Для более подробной информации нажимается ПКМ на объекте. В окне свойств объекта отображаются все имеющиеся у него на данный момент времени свойства. В интерфейс встроен таймер, останавливающийся во время паузы.

**Unit`ы**

Unit`ы обозначены по системе условного обозначения войск НАТО. Объекты активного игрока обозначены красным цветом. Реализована анимация во время атаки. Выделение unit`ов показывается границей зеленого цвета вокруг объекта.

**Реализация**

За игровой интерфейс отвечает проект SummerGame. Графическая сторона построена на системе WPF и размечена на XAML с применением графических изображений. Взаимодействие с моделью игры происходит при помощи привязки данных. Разметка каждого отдельного элемента интерфейса реализована с помощью шаблонов данных и хранится в соответствующих ресурсах.

**Вывод**

Построена концепция игровой модели. Реализована логика игры, пользовательский интерфейс к нему. Налажены взаимодействие графического интерфейса с игровой моделью, промежуточная система пересылки данных из модели в интерфейс, взаимодействие окон друг между другом. Написана модель игрового поля, ведения на нем боевых действий. Реализована многопользовательская система выбора сторон. Боевые единицы наделены возможностью перемещаться, атаковать в зависимости от свойств, меняющихся в ходе боевых действий. Написана модель карты с ландшафтами, влияющими на перемещение боевых единиц, модель игрового времени, в течение которого выполняются приказы, отданные боевым единицам.

Всего на данный момент времени в игре задействованы 5 проектов: AbstractGameEngine, GameEngine, Geometry, SummerGame, UserInterface. За логику игры отвечают проекты: AbstractGameEngine, GameEngine, Geometry. Интерфейс игры реализован в проекте SummerGame. Промежуточная система взаимодействия логики игры с пользовательским интерфейсом содержится в проекте UserInterface.

Проект AbstractGameEngine реализован для абстрактной модели какой бы то ни было игры-стратегии. На его базе можно построить любую игру, пользуясь классами, им предоставляемыми.

GameEngine отвечает за конкретную игру. В данном случае, является главным проектом, отвечающим за логику игры.

В проекте Geometry делаются расчеты для формирования действий Unit`a. В частности, формирование пути через различные области карты, расчет расстояний пути по конкретной области, точки входа и выхода из области.

В промежуточной системе взаимодействия логики игры с пользовательским интерфейсом, UserInterface написаны специальные конверторы-функции для перевода данных игры в необходимые для использования при построении пользовательского интерфейса, такие, как, например, перевод игровых координат в экранные.

Проект SummerGame является финальным в том плане, что представляет собой конечный результат того, какой пользователь увидит игру. Здесь описываются объекты с точки зрения графического интерфейса, получая необходимые данные для его построения из игровой модели через взаимосвязь с проектом UserInterface. Также проект содержит описания и реализацию меню интерфейса и игрового меню.

В итоге, все цели и задачи, поставленные изначально перед разработчиками, были выполнены. В каждой из них были выделены дополнительные подзадачи, исправлены недочеты и добавлены улучшения. Задачи были поставлены и выполнены в ограниченном временном промежутке. Исходя из этого, можно сказать, что проект требует наработок, дополнительной функциональности и обнаружения и исправления существующих недочетов.