федеральное государственное бюджетное образовательное учреждение высшего образования

**«Вологодский государственный университет»**

Институт математики, естественных и компьютерных наук

Кафедра автоматики и вычислительной техники

**КУРСОВАЯ РАБОТА**

Дисциплина: «Проектирование программного обеспечения»

Наименование темы: «Разработка интерактивной компьютерной игры»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 09.03.04.  код направления  подготовки/  специальности | 43.10.  код выпускающей  кафедры | 22  регистрационный номер по журналу | 01  код формы  обучения | 2022  год |

|  |  |
| --- | --- |
| Руководитель | доц. Кочкин Дмитрий Валерьевич |
| Выполнил (а) студент | Самойлов Максим Алексеевич |
| Группа, курс | 4Б09 РПС-21, 2 курс |
| Дата сдачи | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Дата защиты  Оценка по защите | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Вологда

2022 г.

ВВЕДЕНИЕ

Цель курсовой работы

Целью курсовой работы является разработка алгоритмов, кодирование и отладка программного продукта в среде программирования Game Maker Studio 2.

Результат курсовой работы

Результатом работы является исходный текст программы, система тестов для доказательства ее соответствия функциональным требованиям и пояснительная записка, содержащая все необходимые пояснения по этапам разработки программного продукта.

Содержание пояснительной записки

• анализ требований к программному продукту;

• описание алгоритмов и структур данных;

• обоснование архитектуры программы;

• подробное описание функций;

• результаты тестирования программы.

ПОЯСНИТЛЬНАЯ ЗАПИСКА

Анализ требований к программному продукту

1. Корректное вычисление формул, исполняемых функциями
2. Соответствие исполняемых функций с их пользовательским описанием
3. Эффективное хранение данных пользователя

Описание алгоритмов и структур данных

**Алгоритм поведения объектов в сражении**

Поскольку во время хода персонажа все объекты в пространстве не должны продолжать отсчёт до своего хода, была введена булевая глобальная переменная «global.timeflow». При значении false все объекты должны остановить свой отсчёт. Это позволяет игроку иметь безграничное время для выбора своего действия, а также наблюдать визуальное отображение действия персонажа или противника.

После окончания любого действия время должно быть возвращено. Для этой цели объекту-эффекту был добавлен функционал вовзращения времени после окончания непосредственно визуального отображения эффекта. Поскольку каждое действие создаёт экземпляр объекта-эффекта, то можно гарантировать возврат времени.

**Структура данных информации о пользователе**

Сохранение пользователя представляет из себя текстовый документ формата .ini, хранящий в себе пары «ключ – значение».

Файл содержит в себе расположение персонажа, его характеристику и найденное снаряжение.

Расположение файла: C:\Users\*\*Имя Пользователя\**\AppData\Local\StelVis

Обоснование архитектуры программы

Для создания программы была выбрана «Событийно-ориентированная архитектура», заключающаяся в создании событий (определяются как существенные изменения состояния) и реакции на них.

Источниками событий (агентами) являются действия пользователя или временные периоды.

Потребителями событий (стоками) являются объекты приложения.

Создание приложения в рамках архитектуры, управляемой событиями, позволяет программному продукту быть сконструированным способом, способствующим лучшей интерактивности, поскольку системы, управляемые событиями, по структуре более ориентированы на непредсказуемые и асинхронные окружения.

Интерактивная компьютерная игра представляет из себя непредсказуемое и асинхронное окружение, поскольку совершенно неизвестно, как будет вести себя пользователь, и каждое из его действий в программе могут быть выполнены в отличных временных периодах.

Подробное описание функций

Далее приведено подробное описание ключевых функций, используемых в программном продукте.

**Функция move\_choose(id,mana,maxmana,hp,maxhp,manareg):**

**Принимаемые аргументы:**

id – указатель на экзмепляр объекта (аналогично указателю ‘this’ в других языка программирования);

mana – текущее значение энергии экземпляра объекта;

maxmana – максимальное значение энергии экземпляра объекта;

hp – текущее значение здоровья экземпляра объекта;

maxmana – максимальное значение здоровья экземпляра объекта;

manareg – текущее значение скорости восстанавления энергии экземпляра объекта.

**Возвращаемое значение:** move – номер выбранного запланированного действия для врага.

**Алгоритм работы:**

По текущему состоянию аргументов рассчитывается множитель стоимости энергии для использования действия врага, а также численное значение энергии, которой будет обладать объект ко времени исполнения действия.

В зависимости от количества расчитанной энергии выбирается один из нескольких видов действий: поочерёдно проверяется, имеет ли объект доступ к действию, и будет ли у него мана на это действие; иначе проверяется следующее действие по приоритету. Исключением в приоритете будет являться пропуск «массовой атаки», если объект обладает указателем на «цель» действия, то есть вынужден атаковать заданный объект.

Порядок приоритета: Сильная атака – Массовая атака – Обычная атака – Ослабленная атака – Бездействие.

Если была выбрана немассовая атака, выбирается случайным образом один из двух объектов-персонажей, обладающий положительным количеством здоровья, если объект не обладал указателем на «цель» действия, то есть был вынужден атаковать заданный объект.

**Функция draw\_pie(x ,y ,value, max, colour, radius, transparency):**

**Принимаемые аргументы:**

x – координата оси абцисс в центре круга;

y – координата оси ординат в центре круга;

value – текущее значение;

max – максимальное значение;

colour – цвет круга;

radius – радиус круга;

transparency – прозрачность изображения (от 0 – полностью непрозрачный, до 1 – полностью прозрачный).

**Не имеет возвращаемое значение.**

**Алгоритм работы:**

Изображение круга создаётся путём его разбиения на сектора. Соотношение секторов, необходимых для вывода, к общему чилу секторов в круге высчитывается по отношению текущего значения к максимальному.

Создаётся квадратная поверхность для отрисовки курга. Длина поверхности равна диаметру круга.

В созданной поверхности задаются цвет и прозрачность.

Вывод каждого сектора круга осуществляется заданием вершин «треугольников» (общая вершина в центре и обход по контуру круга для остальных вершин), координаты вершин высчитываются по вектору-радиусу.

**Событие Mouse Left Pressed для объекта enemy:**

**Алгоритм работы:**

Событие исполняется, когда была нажата левая кнопка мыши на объект.

Алгоритм проверяет доступность выбранного действия (определеяется в переменной global.spell): если действие не было выбрано, или в текущий момент выведено информационное окно, или действующий персонаж не обладает нужным количеством энергии, то событие ничего не совершает.

Если же действие подходит, оно исполняется.

В зависимости от действия может происходить следующее: создание объекта-эффекта действия, расчёт теряемого здоровья, энерегии, готовности объекта к действию, изменение известной информации об объекте для игрока, а также вычитается треубемое значение энергии у активного персонажа, изменяются переменные-состояния для объекта.

Каждое действие главного персонажа (global.spell от 1 до 20) также имеет дополнительные свойства, которые исполняются при подходящем условии (экипировано снаряжение, меняющее свойства этого действия). Например, добавление новых переменных-состояний для врага.

В конце алгоритма обнуляется указатель на активного персонажа и выбранное действие.

**Событие Draw для объекта dude:**

**Алгоритм работы:**

Событие исполняется каждый программный тик на этапе отрисовки изображения.

Алгоритм выводит изображение самого объекта, его значения готовности, здоровья и энергии, а также выводит отображение состояний, действующих на объекте: каждое последующее отображение будет выведено ниже предыдущего благодаря переменной-счётчику.

Также выполняется проверка на положительное количество очков здоровья персонажа. Если оно оказывается неположительным, цвет изображения персонажа изменяется на более затемнённый, а также проверяется наличие второго персонажа и его здоровье. В случае его отсуствия или неположительного значения, создаётся объект-переход в другое пространство (сражение проиграно, переход в комнату «lose»).

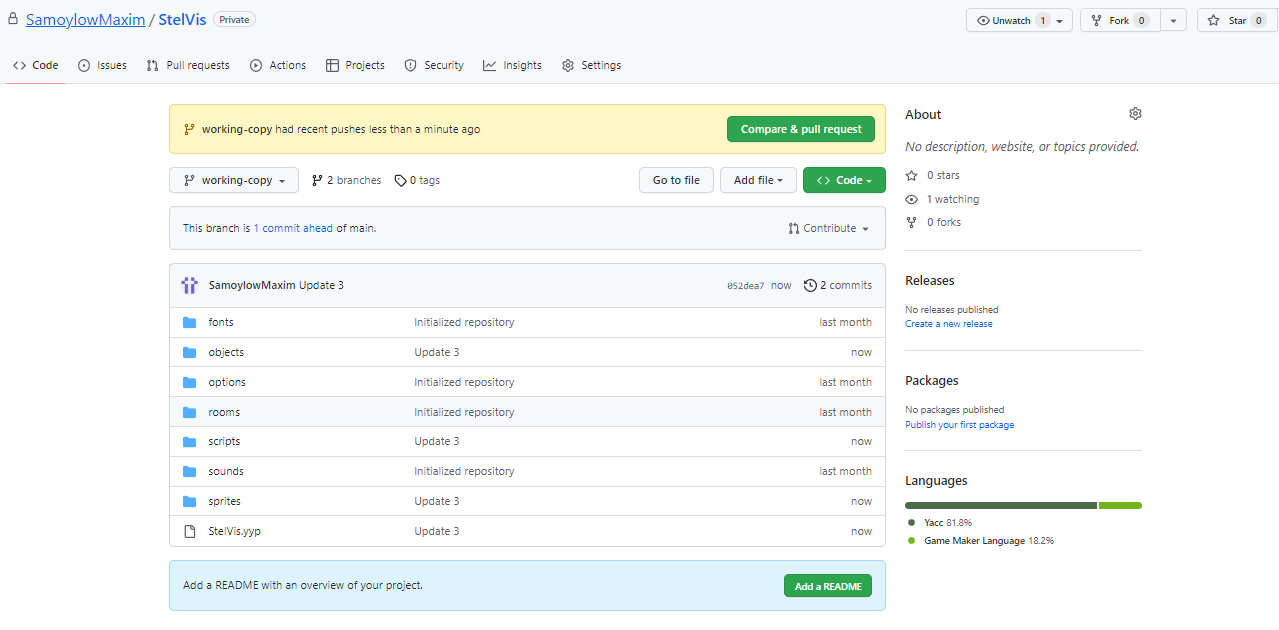
К тому же значение энергии, здоровья и готовности персонажа к действию увеличивается в соотвествии с хранимой скоростью увеличения, если оно не достигло максимального значения, здоровье персонажа положительно, и время не было остановлено (global.timeflow = true).

При достижении нужного количества «готовности» (cooldown >= 100), время останавливается, вырисовывается постепенно расширяющийся и «растворяющийся» круг (визуальное отображение готовности для игрока), уменьшается длительность определённых состояний персонажа при их наличии, передаётся указатель на персонажа в глобальную переменную (global.active), и создаются объекты кнопок для выбора действий, расположенных по правой части эллипса, где центром является персонаж.

Также выводятся дополнительные сведения, если включён режим разработчика: численное значение здоровья, энергии, готовности и их скорости.

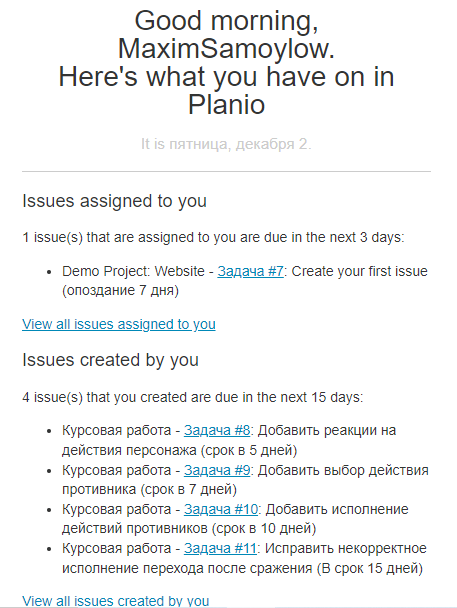
Использование системы контроля версий

В ходе разработки продукта была использована система контроля версий GitHub. Данная система позволяет хранить рабочие и резервные копии продукта для безопасной работы над программой.

*  
Рисунок 1. Использование системы контроля версии*

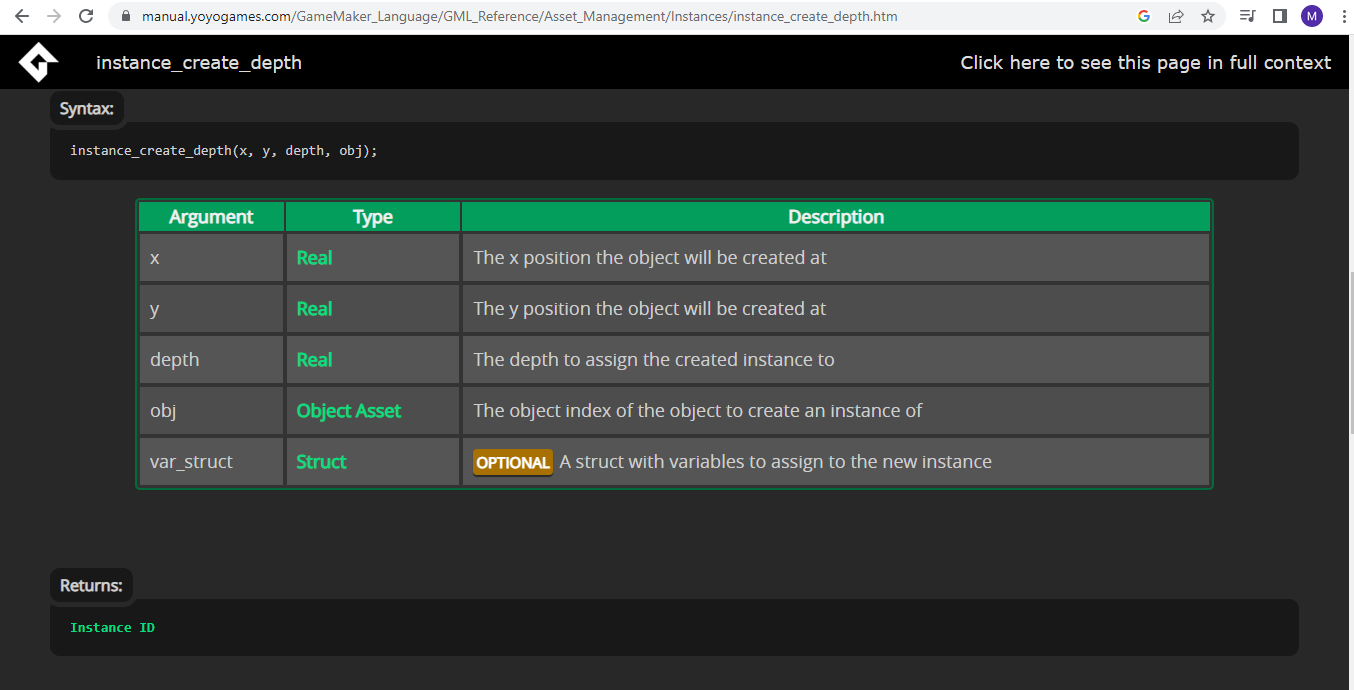
Использование системы отслеживания ошибок

В ходе разработки продукта была использована система отслеживания ошибок Planio. Данная система позволяет планировать разработку программы, составляя отдельные задачи с заданными сроками.

*  
Рисунок 2. Использование системы отслеживания ошибок*

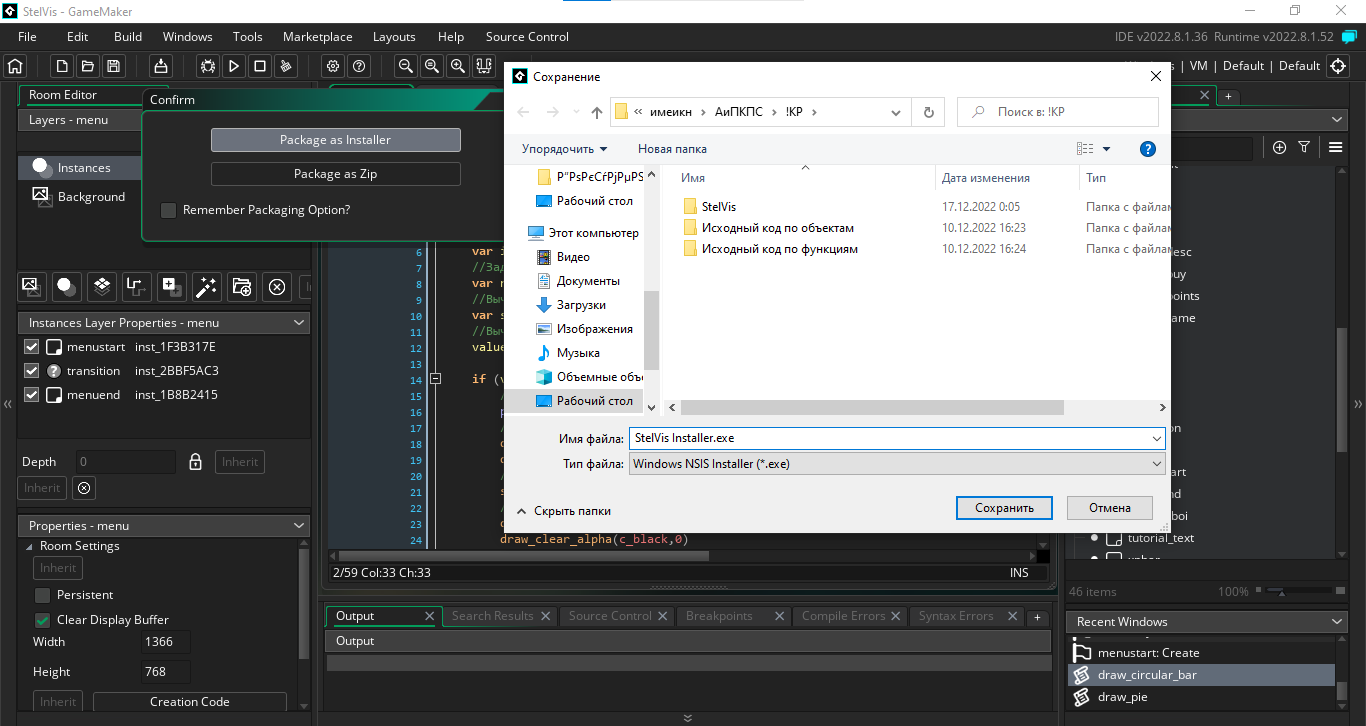
Создание документации приложения

Среда программирования Game Maker Studio 2 не имеет возможности автоматической документации функций, однако обладает официальной документацией по всем стандартным функциям, которые были использованы в программе (manual.yoyogames.com).

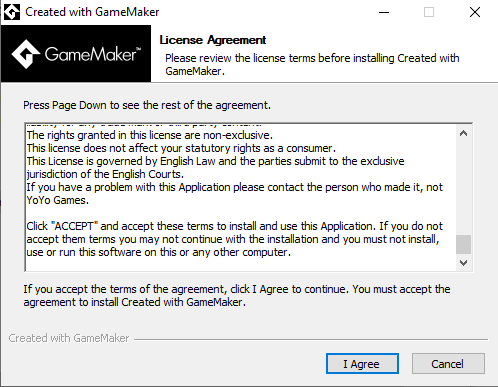
*  
Рисунок 3. Официальная документация функций*

Создание дистрибутива приложения

Среда программирования Game Maker Studio 2 предоставляет возможность компиляции приложения в установачный файл для его дальнейшей установки пользователями.

*  
Рисунок 4. Создание дистрибутива программы*

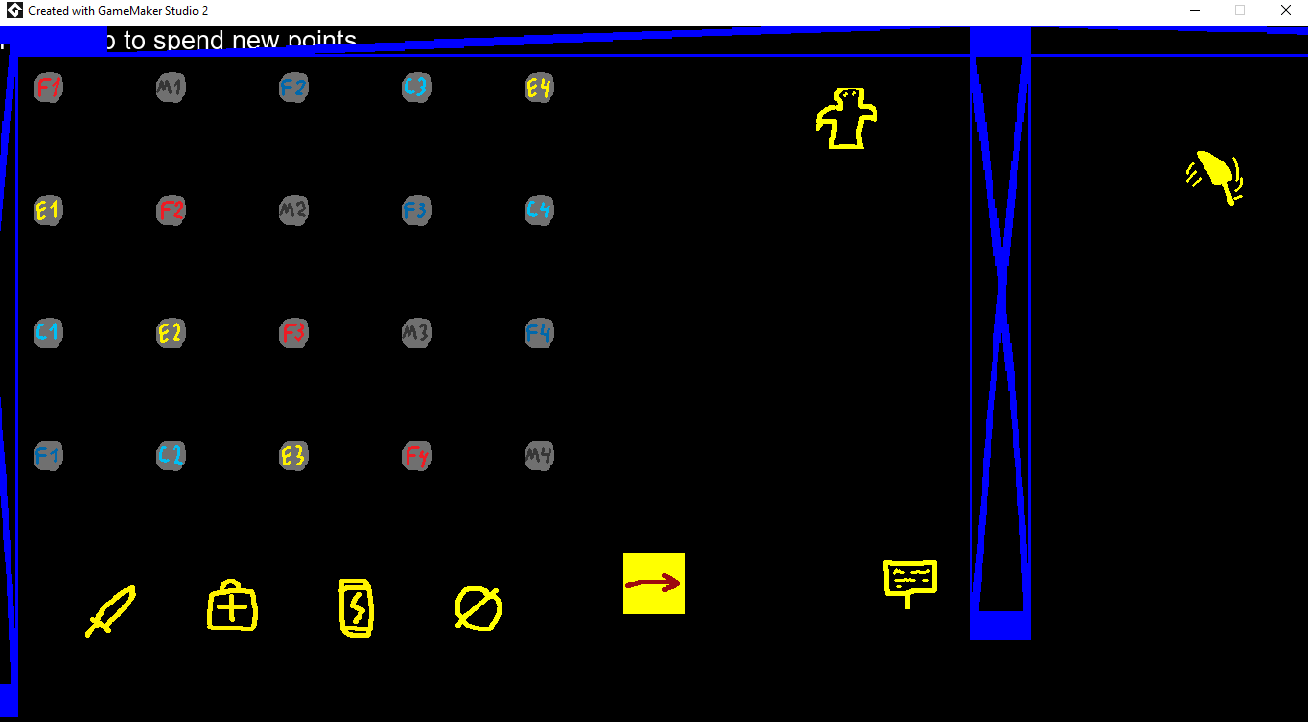
Полученный дистрибутив можно использовать пользователем для установки приложения.

 *Рисунок 5. Установка приложения*

Результаты тестирования программы

Для доказательства соответствия функциональным требованиям была создана система тестов.

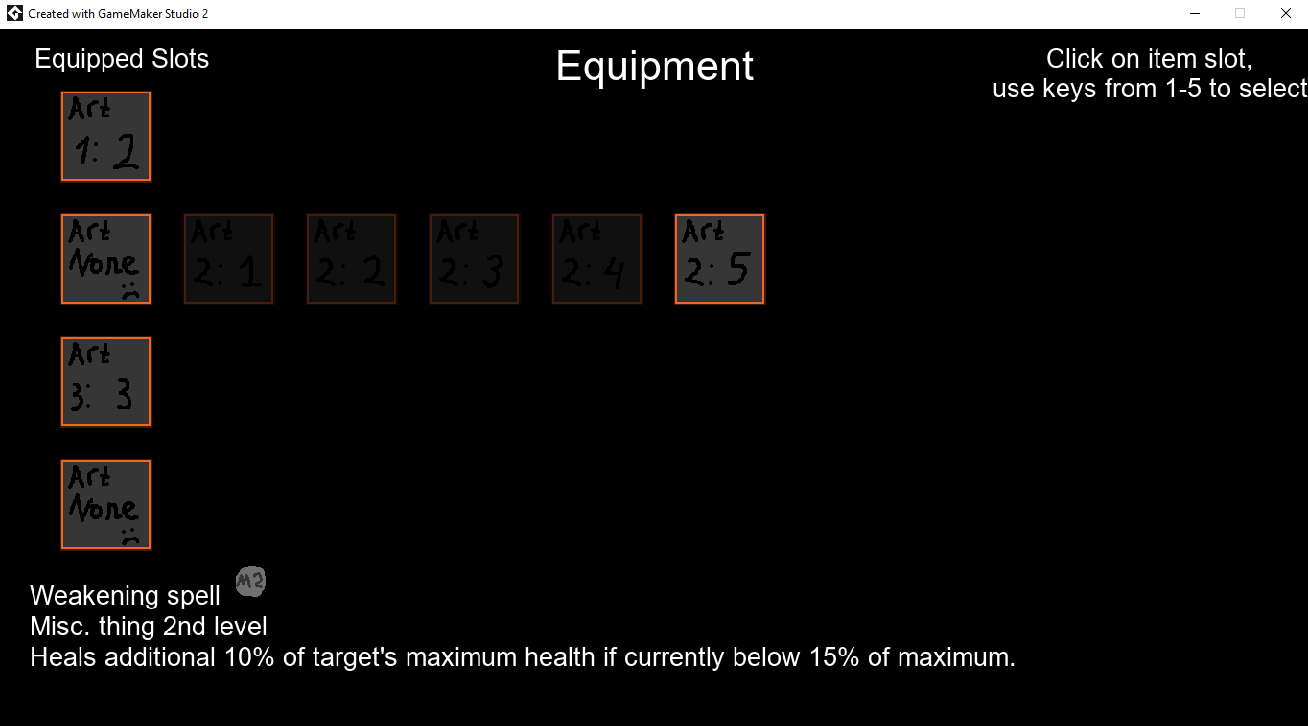
Во время тестирования имеется возможность воссоздать все функции, используемые в программе: взаимодействие с объектами и противниками, развитие персонажа, смена экипировки, сражение, обучение и сохранение прогресса.

  
*Рисунок 6. Помещение для системы тестирования*

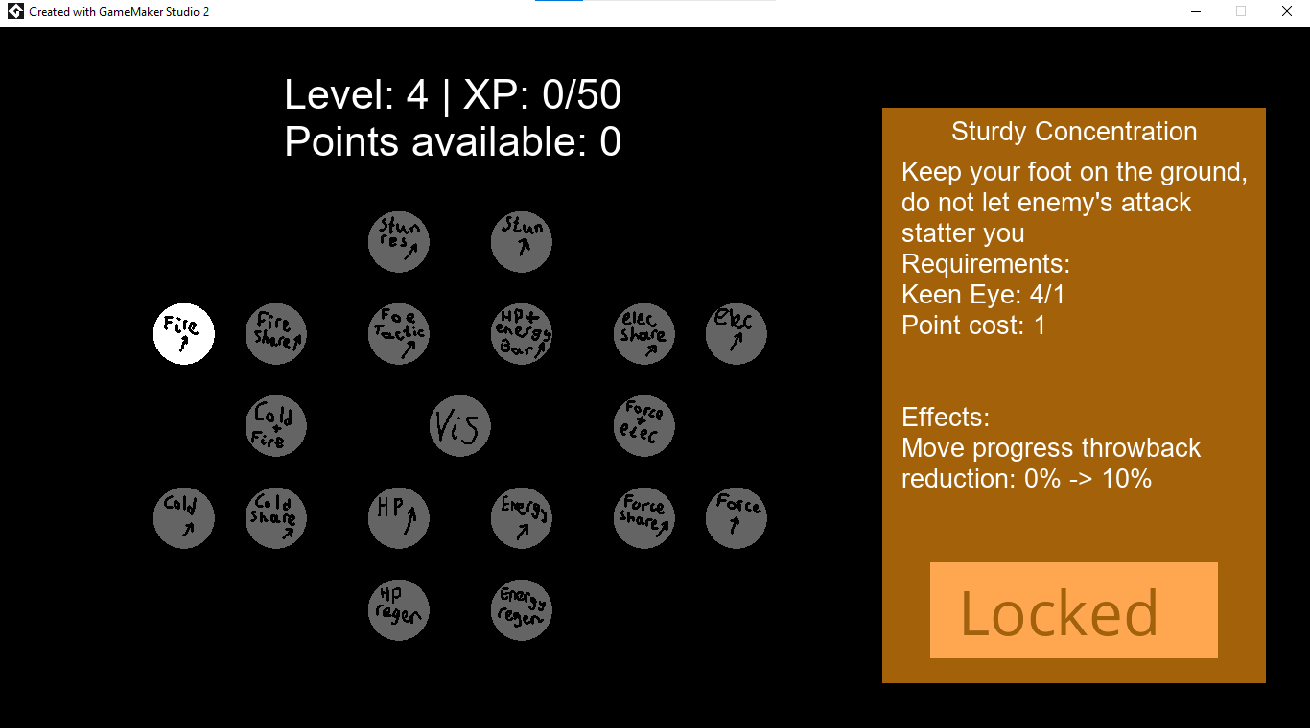
В результате проведённого тестирования была доказана корректность исполнения всего функционала программы.

При подборе предмета он пропадают из комнаты, а его переменная-флаг меняется на истину.

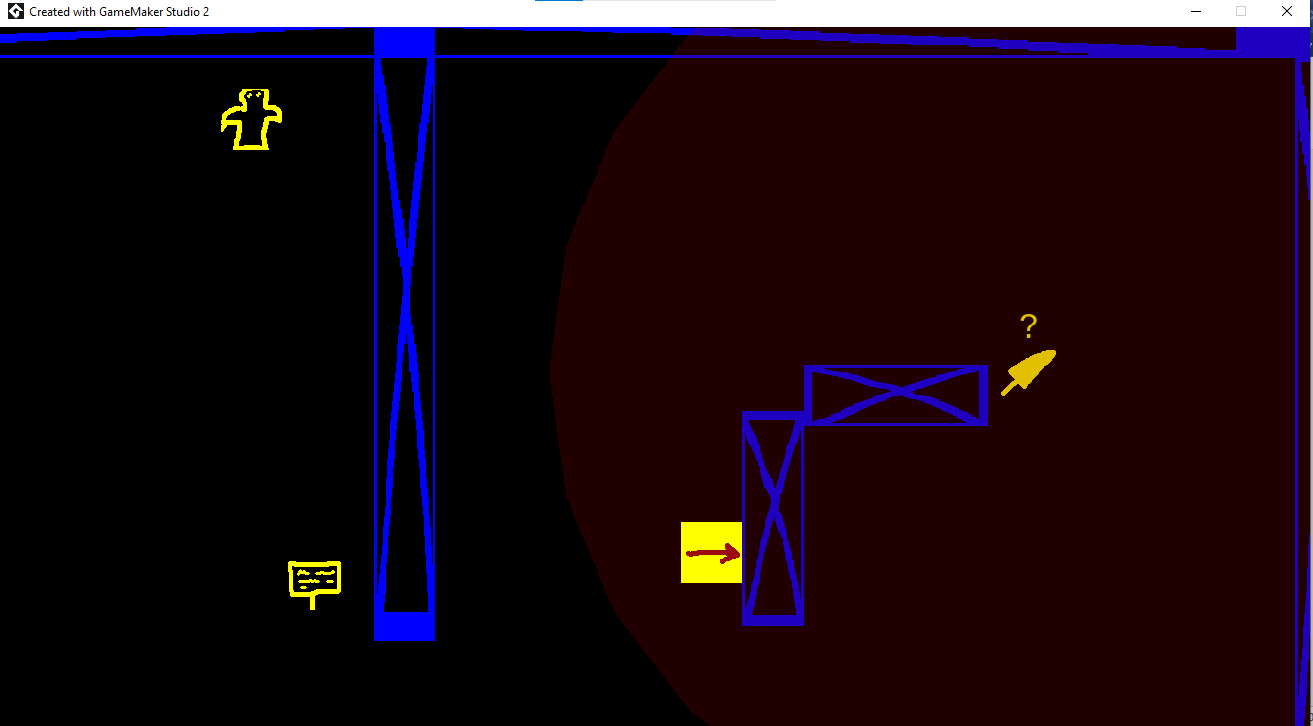
  
*Рисунок 7. Подбор предметов*

В меню экипировки (Доступно при нажатии клавиши I) корректо отображаются подобранные предметы, их описание и предоставляется возможность экипировать их.  
*Рисунок 8. Экипировка оснащения*

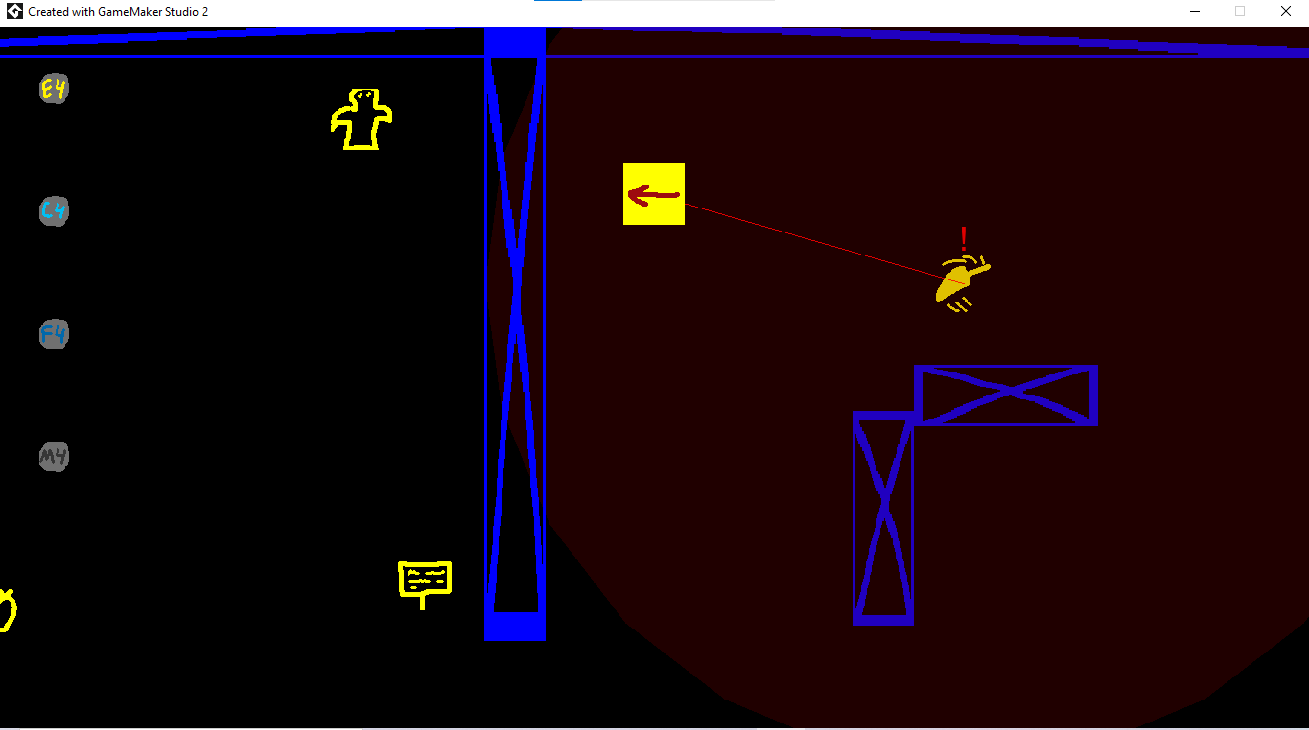
Система развития персонажа (появляется при нажатии клавиши Tab) верно отображает возможные улучшения, их требования и изменения, которые они дают.

  
*Рисунок 9. Развитие персонажа*

При взаимодействии с врагом был использован специально созданный «режим разработчика» (переключается нажатием клавиши Q) для отображения зоны видимости противника.

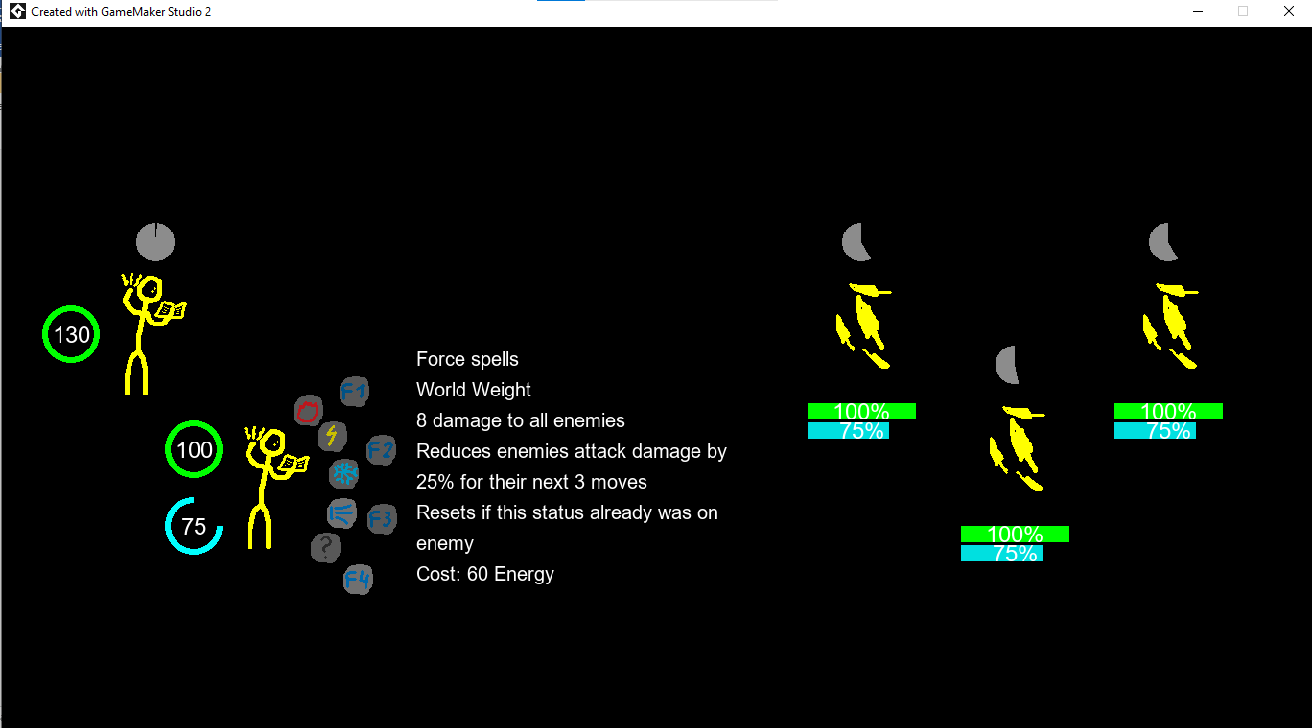
  
*Рисунок 10. Видимость противника*

Противник успешно преследует персонажа, пока тот находится в прямой видимости и попадает в радиус видимости.

  
*Рисунок 11. Преследование персонажа*

При достижении противником персонажа, осуществляется переход в комнату сражения.

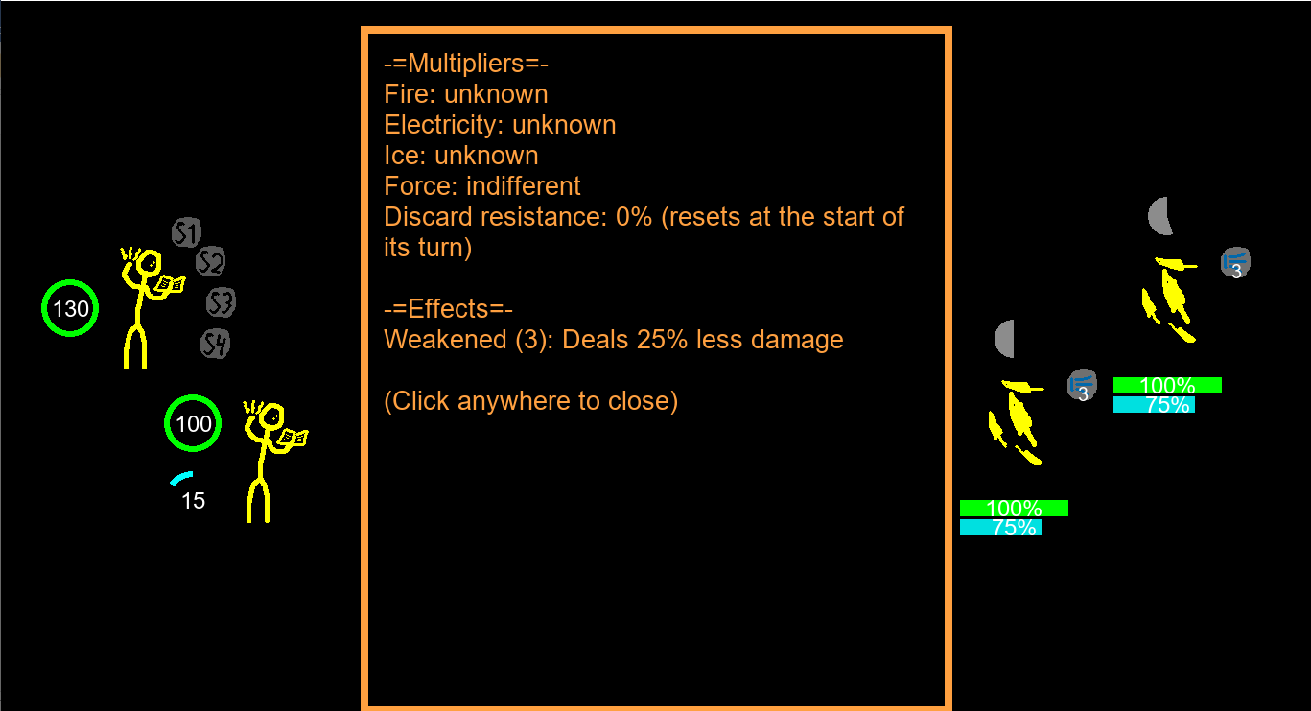
Выполняемые действия персонажа совпадают с их пользовательским описанием и не вызывают ошибок.

 *Рисунок 12. Выбор действия персонажа*

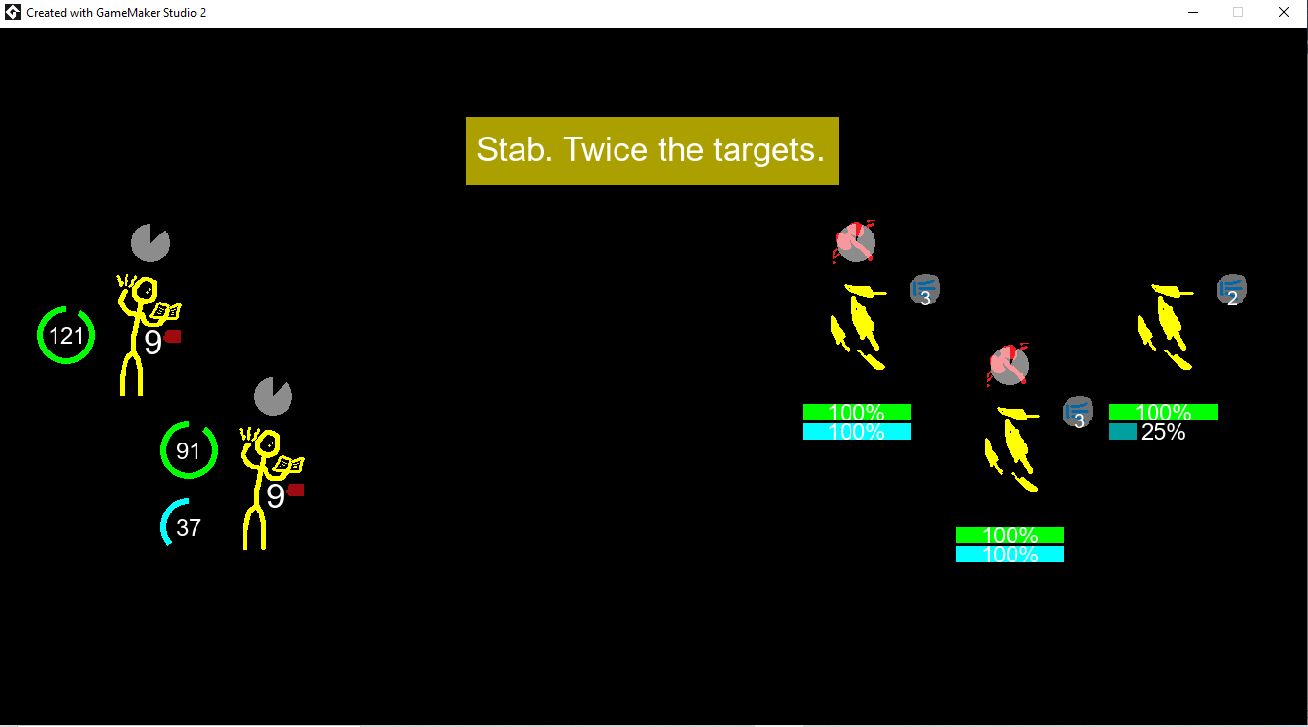
При исполнении действия корректо продолжается ход сражения.

*Рисунок 13. Исполнение действия персонажа*

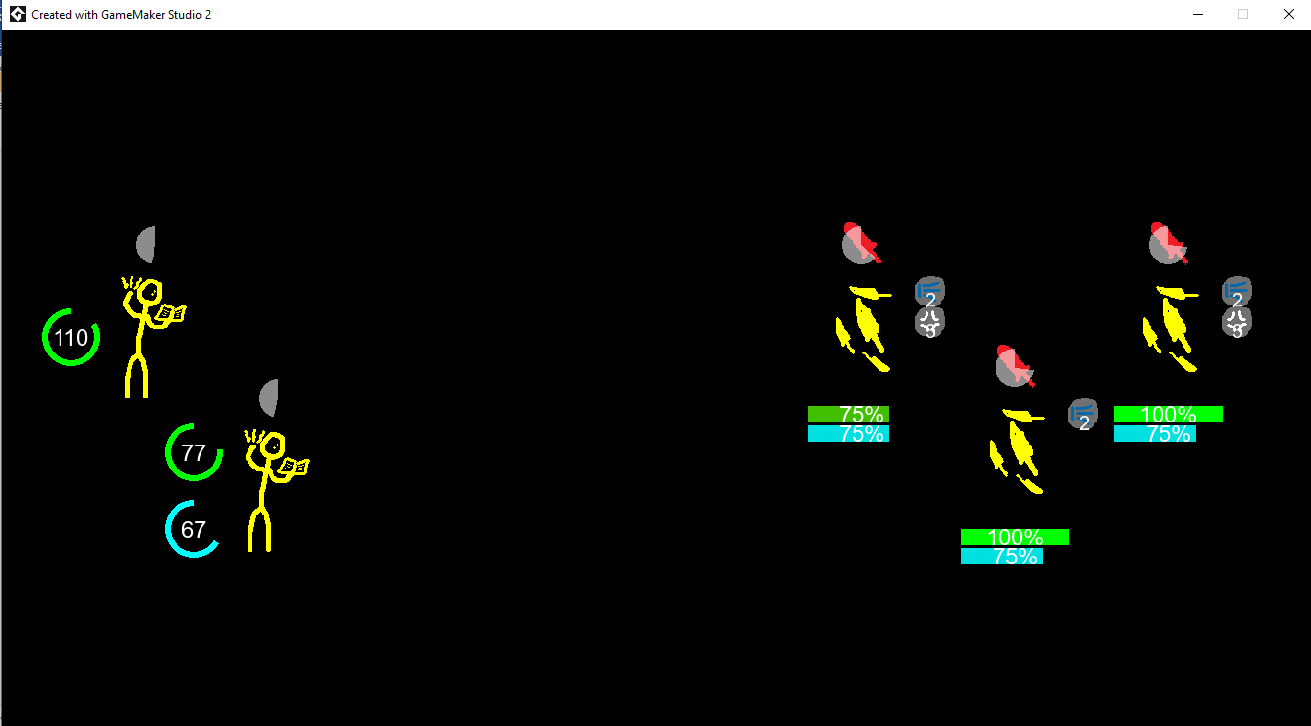
Функция вывода информации о противнике также корректо исполняется.

 *Рисунок 14. Информация о противнике*

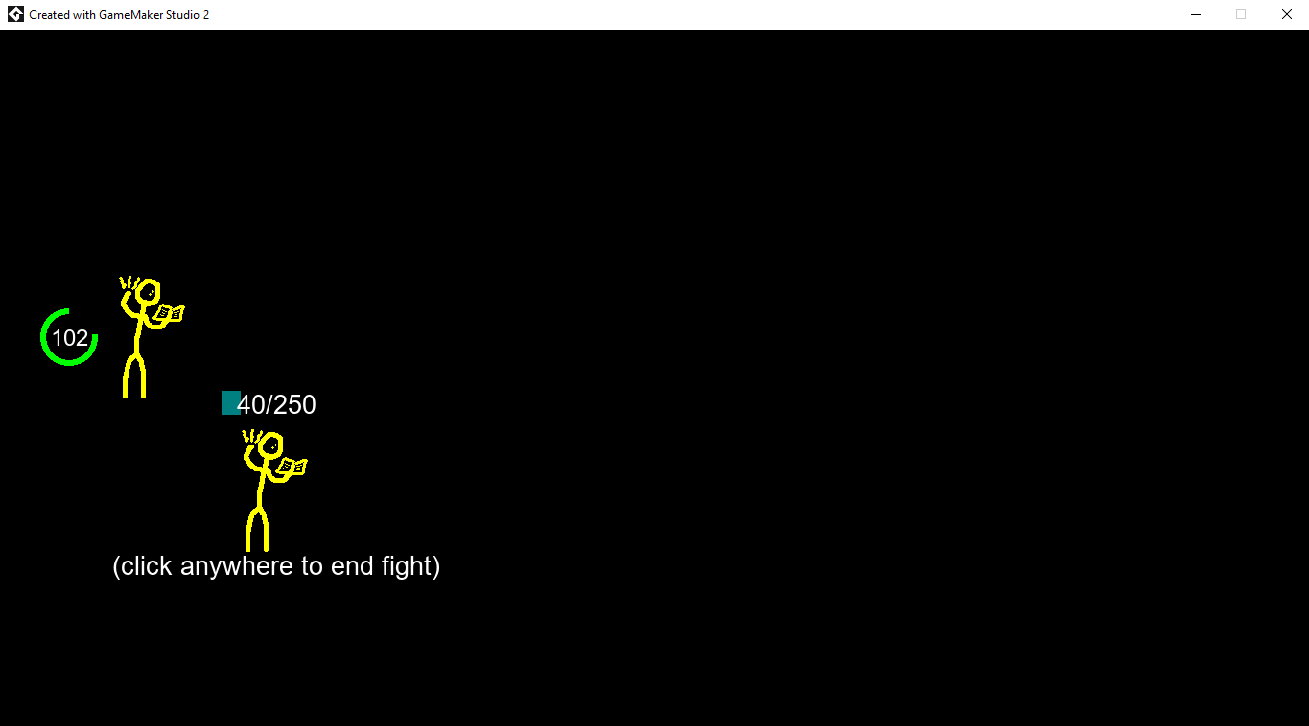
Исполнение действий противников не вызывают сбоев и совпадают с ожидаемым поведением.

  
*Рисунок 15. Действие противника*

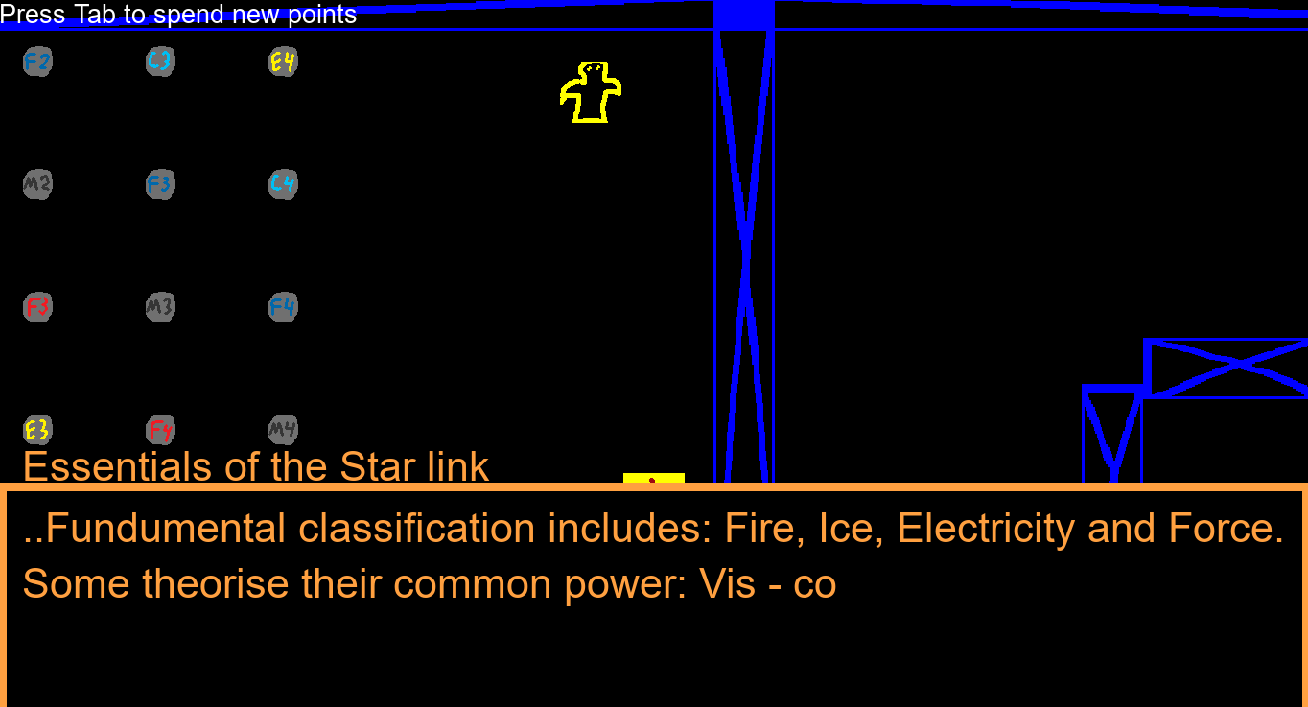
Визуальное отображение состояний и планируемых действий противников работает исправно.

 *Рисунок 16. Отображение состояния противников*

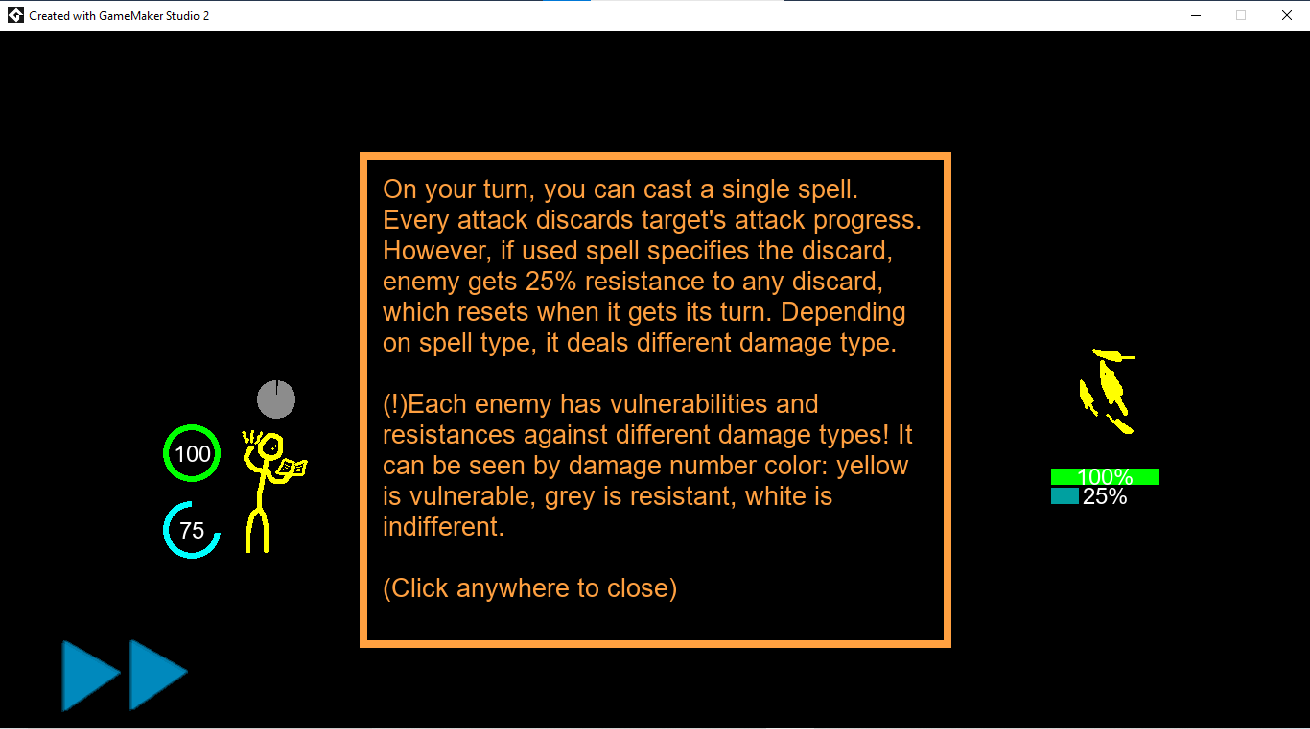
После окончания сражения полоска опыта заполняется в соотвествии с заданными параметрами, и можно успешно вернуться в комнату, откуда был вызван переход в сражение.

  
*Рисунок 17. Окончание сражения*

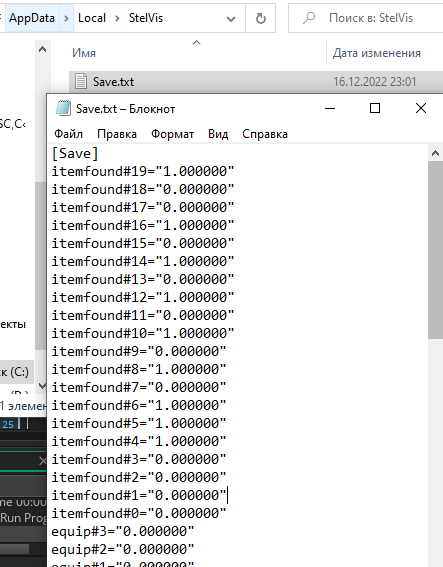
Взаимодействие с объектами корректо выводит текстовое окно, и сам текст обладает ведёт себя по верному алгоритму.

  
*Рисунок 18. Взаимодействие с объектами*

Тренировачное сражение содержит вовремя отображающиеся обучающие тексты.

  
*Рисунок 19. Вступительное обучение*

Процесс выхода из игры успешно создаёт файл сохранения, который будет считываться при следующем запуске для восстановления всего прогресса.

  
*Рисунок 20. Расположение и содержание файла сохранения*

Таким образом, созданная система тестирования позволяет убедиться в работоспособности программы и её соответсвии к требованиям продукта.

ИСХОДНЫЙ ТЕКСТ ПРОГРАММЫ

Каждый объект в программе имеет реакции на события, при которых исполняются части его кода. Далее приведены все виды событий, используемые объектами.

Create – Моментально исполняется экземпляром объекта при его создании.

Step – Исполняется экземпляром объекта каждый «шаг» программы.

Alarm 0..9 (Alarm[0..9]) – Исполняется экземпляром объекта после отсчёта некоторого количества «шагов» программы.

Draw – Исполняется экземпляром объекта каждый «шаг» программы во время отрисовки.

Left/Right Pressed – Исполняется экземпляром объекта, когда была нажата левая/правая кнопка мыши по нему.

Key Press – Исполняется экземпляром объекта, когда была нажата заданная клавиша.

Mouse Enter/Leave – Исполняется экземпляром объекта, когда курсор мыши входит/выходит в его пространство.

Destroy – Исполняется экземпляром объекта перед его удалением.

Animation End – Исполняется экземпляром объекта при окончании цикла его анимации.

Collision – Исполняется экземпляром объекта, когда он соприкасается с заданным объектом.

Room Start – Исполняется экземпляром объекта, когда инициализируется переход к пространству (комнате), если он находится в нём.

**Объект главного персонажа в сражении**

**Событие Create:**

//Глобальная переменная, хранящая указатель на текущего персонаж

//(ключевое слово "noone" означает пустое значение)

global.active = noone;

//Проверка, поставлена ли игра на паузу

paused = false;

//Максимальное значение здоровья

maxhp = round(100\*(1+0.2\*global.hpbuff));

//Максимальное значение "маны"

maxmana = round(100\*(1+0.2\*global.manabuff));

//Установление текущего здоровья

hp = maxhp;

//Установление текущей маны

mana = round(maxmana/2);

//Переменная для отсчёта времени, когда настанет ход этого персонажа (до 100)

cooldown = 0;

//Переменная, хранящая количество действий, которое может совершить персонаж

act = 0;

//Переменная для задания радиуса круга

cirradius = 0;

//Массивы хранят Множитель для формул; Количество ходов, которые он действует

//Ускорение хода

CDbuff = [1,0];

//Замедление хода

CDdebuff = [1,0];

//Усиление атак

Forcebuff = [1,0];

//Усиление урона всех атак

DMGbuff = [1,0];

//Ослабление урона всех атак

DMGdebuff = [1,0];

//Разовое усиление заклинаний

meditatebuff = 0;

//Выбранное заклинание

global.spell = 0;

//Задание случайного "seed"

randomize();

//Выбор врагов на основе значения глобальной переменной

//Создаёт разное количество врагов, задаёт им их разновидность и отсчёт времени до их кода "alarm", задаёт получаемый опыт после битвы

switch (global.encounter) {

case 0:

var inst = instance\_create\_depth(1152,384,0,enemy);

inst.metype = 0;

inst.alarm[0] = 2;

global.encounterxp = 50;

break;

case 1:

var inst = instance\_create\_depth(896,320,0,enemy);

inst.metype = 1;

inst.alarm[0] = 2;

var inst = instance\_create\_depth(1056,448,0,enemy);

inst.metype = 1;

inst.alarm[0] = 2;

var inst = instance\_create\_depth(1216,320,0,enemy);

inst.metype = 1;

inst.alarm[0] = 2;

global.encounterxp = 0;

break;

}

//Глобальная переменная для повторной атаки определённого заклинания

global.forcerepeat = false;

//Если Первый слот экипировки занят артефактом №4

if global.equip[0] == 4 {

//Определённое заклинание получает повторную атаку

global.forcerepeat = true;

}

**Событие Alarm[0]:**

//Если не существует инстанции объекта "текст обучения"

if !instance\_exists(tutorial\_text){

//Глобальная переменная, хранящая, течёт ли время, устанавливается на "правда"

//(т. е. время начнёт идти)

global.timeflow = true;

}

**Событие Draw:**

//Выводит на экран собственное изображение

draw\_self();

//Если текущее здоровье достигло 0

if (hp <= 0) {

//Установливается цвет изображения как более затемнённый

image\_blend = make\_color\_hsv(0,0,150)

//Если существует второй персонаж

if instance\_exists(dude2) {

//И если у него текущее здоровье также достигло 0, то битва проиграна

if dude2.hp <= 0 {

//Если не существует инстанция объекта "переход"

if !instance\_exists(transition) {

//"Переход" создаётся

var inst = instance\_create\_depth(x,y,-3,transition);

//Его переменная goto задаётся как комната "lose"

inst.goto = lose;

}

}

//Иначе персонаж всего один, и битва проиграна

} else {

//Аналогичный "переход" в комнату "lose"

if !instance\_exists(transition) {

var inst = instance\_create\_depth(x,y,-3,transition);

inst.goto = lose;

}

}

}

//Иначе персонаж жив, можно выводить интерфейс битвы

else {

//Но только если не существует инстанции объекта "полоска опыта",

//которая бы означала конец битвы

if !instance\_exists(xpbar){

//Перед выводом изображений и текста до них задаются их параметры:

//draw\_set\_alpha задаёт непрозрачность [от 0 до 1]

//draw\_set\_font задаёт шрифт для текста

//draw\_set\_color задаёт цвет для текста

//draw\_set\_halign задаёт выравнивание текста по горизонтали (слева, по центру, справа)

draw\_set\_alpha(1)

draw\_set\_font(font\_gui)

//Вывод изображения круга слева от персонажа, показываюшая значение здоровья

draw\_circular\_bar(x-sprite\_width/2-48,y-sprite\_height/2+24,hp,maxhp,c\_lime,30,1,6)

draw\_set\_color(c\_white)

draw\_set\_halign(fa\_center)

//Вывод текста слева от персонажа, показываюшая значение здоровья

draw\_text(x-sprite\_width/2-48,y-sprite\_height/2+12,ceil(hp))

//Вывод изображения круга слева от персонажа, показываюшая значение маны

draw\_circular\_bar(x-sprite\_width/2-48,y+sprite\_height/2-24,mana,maxmana,c\_aqua,30,1,6)

draw\_set\_color(c\_white)

draw\_set\_halign(fa\_center)

//Вывод текста слева от персонажа, показываюшая значение здоровья

draw\_text(x-sprite\_width/2-48,y+sprite\_height/2-36,round(mana))

//Вывод изображения состояний персонажа:

//Переменная-счётчик

var i = 1;

//Если ускорение активно (количество оставшихся ходоов больше 0)

if CDbuff[1] > 0 {

draw\_set\_alpha(1);

//Вывод изображения, с учётом сдвига положения, если до этого были выведены другие состояния

//(благодаря переменной-счётчика можно сдвигать координату на 32 пикселя за каждое другое состояние)

draw\_sprite(s\_status,2,x-sprite\_width/2+32\*(i-2),y+sprite\_height/2+32);

draw\_set\_font(font\_spell\_desc);

draw\_set\_color(c\_white);

//Вывод продолжительности статуса

draw\_text(x-sprite\_width/2+32\*(i-2),y+sprite\_height/2+32,CDbuff[1]);

//Увеличение счётчика-переменной

i++;

}

//Аналогично для остальных состояний, за исключением дополнительных функций

if Forcebuff[1] > 0 {

draw\_set\_alpha(1);

draw\_sprite(s\_status,4,x-sprite\_width/2+32\*(i-2),y+sprite\_height/2+32);

draw\_set\_font(font\_spell\_desc);

draw\_set\_color(c\_white);

//Если Третий слот экипировки занят артефактом №4

if global.equip[2] == 4 {

//Вместо продолжности статуса в ходах, оно выводится в процентах (особое изменение)

draw\_text(x-sprite\_width/2+32\*(i-2),y+sprite\_height/2+32,string(Forcebuff[1])+"%");

} else {

draw\_text(x-sprite\_width/2+32\*(i-2),y+sprite\_height/2+32,Forcebuff[1]);

}

i++;

}

if DMGbuff[1] > 0 {

draw\_set\_alpha(1);

draw\_sprite(s\_status,0,x-sprite\_width/2+32\*(i-2),y+sprite\_height/2+32);

draw\_set\_font(font\_spell\_desc);

draw\_set\_color(c\_white);

draw\_text(x-sprite\_width/2+32\*(i-2),y+sprite\_height/2+32,DMGbuff[1]);

i++;

}

//Статус от медитации не ограничен по длительности, поэтому её продолжительность не выводится

if meditatebuff > 0 {

draw\_set\_alpha(1);

draw\_sprite(s\_status,6,x-sprite\_width/2+32\*(i-2),y+sprite\_height/2+32);

i++;

}

//Если готовность к действию персонажа меньше 100, и если время не остановлено

if cooldown < 100 and global.timeflow {

//Увеличение готовности персонажа

cooldown += 0.2\*CDbuff[0]\*CDdebuff[0];

//Если готовность близка к 100, и существует инстанция объекта "текст обучения"

if cooldown > 99 and instance\_exists(tutorial\_text) {

//Если значение номера подсказки равно 2

if tutorial\_text.tip == 2 {

//Устанавливается отсчёт для "alarm" текста

tutorial\_text.alarm[0] = 1;

}

}

//Если текущая мана меньше максимальной

if mana < maxmana {

//Увеличение значения маны

mana += 0.05\*power(1.1,global.manaregen);

}

//Аналогично со здоровьем

if hp < maxhp {

hp += 0.005\*power(1.15,global.hpregen);

}

//Если готовность достигла 100 и время течёт

if cooldown >= 100 and global.timeflow {

//Если статус ускорения не закончился

if CDbuff[1] > 0 {

//Если Третий слот экипировки занят артефактом №2 (наносит урон врагам от ускорения)

if global.equip[2] == 2 {

//Для каждой инстанции объекта "враг" производится код:

with enemy {

//Если не было известно отношения врага к типу электрического урона,

//делаем его известным (уязвимость, если множитель урона больше 1; сопротивление, если меньше 1; иначе безразличен)

if elecinf == "unknown" {

if elecmod > 1 {

elecinf = "vulnerable"

} else

if elecmod < 1 {

elecinf = "resistant"

} else {

elecinf = "indifferent"

}

}

//Создаётся эффект атаки на себе

var inst = instance\_create\_depth(x,y,-2,effect);

//Этот эффект не возвращаёт время после его окончания

inst.return\_time = false;

//Эффект отображается как электрический

inst.sprite\_index = s\_electric;

//Передаётся указатель на себя

inst.aim = id;

//Передаётся значение получаемого урона

inst.dmg = round((dude.CDbuff[0] - 1)\*global.damage\*elecmod\*harmmod[0])

//Уменьшается собственное здоровье на этот урон

hp -= round((dude.CDbuff[0] - 1)\*global.damage\*elecmod\*harmmod[0])

}

}

//Уменьшение длительности статуса на 1 ход

CDbuff[1] -= 1;

//Если он достиг 0

if CDbuff[1] == 0 {

//Множитель ускорения возвращается к 1

CDbuff[0] = 1;

}

}

//Если статус замедления не закончился

if CDdebuff[1] > 0{

//Уменьшение длительности статуса на 1 ход

CDdebuff[1] -= 1;

//Если он достиг 0

if CDdebuff[1] == 0 {

//Множитель ускорения возвращается к 1

CDdebuff[0] = 1;

}

}

//Если статус усиления особых заклинаний не закончился, и Третий слот экипировки НЕ занят артефактом №4

if Forcebuff[1] > 0 and global.equip[2] != 4 {

//Уменьшение длительности статуса на 1 ход

Forcebuff[1] -= 1;

//Если он достиг 0

if Forcebuff[1] == 0 {

//Множитель ускорения возвращается к 1

Forcebuff[0] = 1;

}

}

//Остановка времени

global.timeflow = false;

//Если объект "ускорение" (для увеличение скорости игры) НЕпрозрачен, т.е. активен

if spedup.image\_alpha == 1 {

//Возврщаем скорость игры на нормальную (60 кадров в секунду)

game\_set\_speed(60,gamespeed\_fps);

}

//Устанавливаем число доступных действий = 1

act = 1;

//Передаём указатель на себя как текущего персонажа

global.active = id;

//Устанавливаем радиус круга = 1 (для анимации расширения круга)

cirradius = 1

//Создаём кнопки для выбора типа заклинаний по эллипсу

for (var i = 1; i<=6; i++){

//Не выводится последняя шестая кнопка, если не открыт доступ к ней (глобальная переменная vis определяет доступ)

if (global.vis > 0 or i<6) {

//Создание кнопки

instance\_create\_depth(x+70\*sin(0.47\*i),y-90\*cos(0.47\*i),-1,skill)

target = instance\_nearest(x+70\*sin(0.47\*i),y-90\*cos(0.47\*i),skill)

//Задание изображения кнопки

target.image\_index = i-1;

//Задание номера кнопки

target.mewho = i;

}

}

}

}

//Выводим прогресс готовности персонажа

draw\_pie(x,y-sprite\_height/2-32,cooldown,100,c\_white,20,0.55-cirradius/50)

//Вывод доп. сведений, если активирован режим разработчика (global.debug)

if global.debug == 1 {

draw\_set\_halign(fa\_center);

var test = string(cooldown)+"+"+string(0.05\*power(1.1,global.manaregen))+". hp+"+string(0.005\*power(1.15,global.hpregen))

draw\_text(x,y-sprite\_height/2-64,test);

}

//Если радиус круга для расширения в диапазоне (0; 30)

if cirradius > 0 and cirradius < 30 {

draw\_set\_color(c\_yellow)

//Вывод круга, постепенно становится прозрачней от увеличения его радиуса

draw\_set\_alpha(1-cirradius/30)

draw\_circle(x,y-sprite\_height/2-32,cirradius,false)

cirradius += 1

draw\_set\_color(c\_white)

}

}

}

**Событие Left Pressed:**

//Если выбрано какое-то заклинание и нет объекта, выводящего информацию о враге

if global.spell > 0 and !instance\_exists(enemy\_info) {

//Задаём переменную отмены заклинания (изначально ложь)

var cancel = false;

//Проверка на выбранное заклинание

//Каждое заклинание проверяет, есть ли достаточно маны на её использование,

//Иначе заклинание отменяется (cancel = true)

//Также каждое заклинание выводит эффект его выполнения с помощью инстанций объекта "effect",

//а ещё убирается бонус от медитации, если таковой был

switch (global.spell) {

//Fire

case 3: //Energy Cauterization

if round(mana) >= 50 {

//Убираются негативные статусы

CDdebuff[0] = 1;

CDdebuff[1] = 0;

DMGdebuff[0] = 1;

DMGdebuff[1] = 0;

//Даётся статус увеличенного урона атак

DMGbuff[0] = 1 + (0.1+(global.damage/100))\*power(1.1,global.fireup);

DMGbuff[1] = round(3\*power(1.1,global.fireup));

mana -= 50;

var inst = instance\_create\_depth(x,y,-2,effect)

inst.sprite\_index = s\_buff;

inst.dmg = 0;

inst.aim = id;

if dude.meditatebuff > 0 {

global.damage -= dude.meditatebuff;

dude.meditatebuff = 0;

}

} else cancel = true;

break;

//Electric

case 7: //Haste Embodiment

if round(mana) >= 30 {

//Даётся статус ускорения

CDbuff[0] = 1 + (0.4+(global.damage/100))\*power(1.1,global.elecup);

CDbuff[1] = round(3\*power(1.1,global.elecup));

mana -= 30;

var inst = instance\_create\_depth(x,y,-2,effect)

inst.sprite\_index = s\_buff;

inst.dmg = 0;

inst.aim = id;

if dude.meditatebuff > 0 {

global.damage -= dude.meditatebuff;

dude.meditatebuff = 0;

}

} else cancel = true;

break;

//Force

case 15: //Robust Stance

if round(mana) >= 30 {

//Даётся статус услинеия особых заклинаний

Forcebuff[0] = 1 + (0.15+(global.damage/100))\*power(1.1,global.forceup);

//Продолжительность определяется по-другому, если экипирован в Третий слот артифакт №4

if global.equip[2] == 4 {

Forcebuff[1] = round(100\*power(1.1,global.forceup));

} else Forcebuff[1] = round(3\*power(1.1,global.forceup));

mana -= 30;

var inst = instance\_create\_depth(x,y,-2,effect)

inst.sprite\_index = s\_buff;

inst.dmg = 0;

inst.aim = id;

if dude.meditatebuff > 0 {

global.damage -= dude.meditatebuff;

dude.meditatebuff = 0;

}

} else cancel = true;

break;

//Misc. (Не требует маны)

case 18: //Tend to Wounds

//Лечит персонажа

var inst = instance\_create\_depth(x,y,-2,effect)

inst.sprite\_index = s\_buff;

inst.dmg = round(maxhp/10);

inst.aim = id;

//Если Второй слот экипировки занят артефактом №5 и здоровье критически мало

if global.equip[1] == 5 and hp <= maxhp\*0.15 {

//Дополнительно лечит больше здоровья

hp += round(maxhp/10);

inst.dmg += round(maxhp/10);

}

hp += round(maxhp/10);

if (hp > maxhp) hp = maxhp;

break;

case 19: //Meditate

//Восстанавливает часть маны

//Если Третий слот экипировки занят артефактом №5

if global.equip[2] == 5 {

//Даётся бонус от медитации, если его ещё не было

if meditatebuff > 0 {

global.damage -= meditatebuff;

}

meditatebuff = global.damage/2

global.damage += meditatebuff;

}

var inst = instance\_create\_depth(x,y,-2,effect)

inst.sprite\_index = s\_buff;

inst.dmg = round(maxmana/4);

inst.aim = id;

mana += round(maxmana/4);

break;

default:

cancel = true;

break;

//Действия, выполняемые вторым персонажем. Некоторые не требуют ману

//NPC #1 - Fighter

case 27: //Tend to Wounds

//Лечит персонажа

var inst = instance\_create\_depth(x,y,-2,effect)

inst.sprite\_index = s\_buff;

inst.dmg = round(maxhp/10);

inst.aim = id;

hp += round(maxhp/10);

if (hp > maxhp) hp = maxhp;

break;

//NPC #2 - Healer

case 30: //Field Medicine

//Усиленно лечит персонажа

var inst = instance\_create\_depth(x,y,-2,effect)

inst.sprite\_index = s\_buff;

inst.dmg = round(maxhp\*0.15);

inst.aim = id;

hp += round(maxhp\*0.15);

if (hp > maxhp) hp = maxhp;

break;

case 31: //nLife Force

//Значительно лечит персонажа

if round(dude2.mana) >= 60 {

dude2.mana -= 60;

hp += global.damage\*3

if (hp > maxhp) hp = maxhp;

var inst = instance\_create\_depth(x,y,-2,effect)

inst.sprite\_index = s\_buff;

inst.dmg = global.damage\*3;

inst.aim = id;

} else cancel = true;

break;

case 32: //Cleansing

if round(dude2.mana) >= 50 {

//Снимает все отрицательные статусы

CDdebuff[0] = 1;

CDdebuff[1] = 0;

DMGdebuff[0] = 1;

DMGdebuff[1] = 0;

dude2.mana -= 50;

var inst = instance\_create\_depth(x,y,-2,effect)

inst.sprite\_index = s\_buff;

inst.dmg = 0;

inst.aim = id;

} else cancel = true;

break;

//NPC #3 - Energiser

case 34: //Tend to Wounds

//Лечит персонажа

var inst = instance\_create\_depth(x,y,-2,effect)

inst.sprite\_index = s\_buff;

inst.dmg = round(maxhp/10);

inst.aim = id;

hp += round(maxhp/10);

if (hp > maxhp) hp = maxhp;

break;

case 35: //nEnergy Flux

if round(dude2.mana) >= global.damage\*6 {

//Передаёт часть маны персонажу

if (mana < maxmana){

dude2.mana -= global.damage\*6;

mana += global.damage\*3

if (mana > maxmana) mana = maxmana;

}

var inst = instance\_create\_depth(x,y,-2,effect)

inst.sprite\_index = s\_buff;

inst.dmg = global.damage\*3;

inst.aim = id;

} else cancel = true;

break;

}

//Конец исполнения заклинания

//Если заклинание не было отменено

if !cancel {

//Удаление всех кнопок

instance\_destroy(skill);

//Возвращение скорости игры на ускоренную, если оно таким было

if spedup.image\_alpha == 1 {

game\_set\_speed(240,gamespeed\_fps);

}

//У активного персонажа отбрасывается готовность, а также уменьшается оставшаяся длительность статусов ослабления и усиления

with (global.active) {

cooldown = 0;

act = 0;

cirradius = 0;

if DMGbuff[1] > 0 and global.spell != 3 {

DMGbuff[1] -= 1;

if DMGbuff[1] == 0 {

DMGbuff[0] = 1;

}

}

if DMGdebuff[1] > 0{

DMGdebuff[1] -= 1;

if DMGdebuff[1] == 0 {

DMGdebuff[0] = 1;

}

}

}

//Обнуление активного персонажа и заклинания

global.active = noone;

global.spell = 0;

}

}

**Событие Right Pressed:**

//Если не существет объекта "информация", а также либо время движется, либо сейчас ход персонажа (есть наличие кнопок "skill")

if !instance\_exists(enemy\_info) and (global.timeflow or instance\_exists(skill)) {

//Время останавливается

global.timeflow = false;

//Создаётся инстанция объекта "информация"

instance\_create\_depth(x,y,-3,enemy\_info);

//Она заполняется текущими состояниями персонажа

enemy\_info.text = "-=Effects=-";

if CDbuff[1] > 0 {

enemy\_info.text += "\nCharged ("+string(CDbuff[1])+"): Increases preperation speed by "+string(round((CDbuff[0] - 1)\*100))+"%";

if global.equip[2] == 2 {

enemy\_info.text += ", (\*)Before action, deal "+string(round(global.damage\*0.5\*power(1.1,global.elecup)))+" damage to all enemies";

}

}

if Forcebuff[1] > 0 {

if global.equip[2] == 4 {

enemy\_info.text += "\n(\*)Robust ("+string(Forcebuff[1])+"%): Increases force spells damage by "+string(round((Forcebuff[0] - 1)\*100))+"% and ignores discards"

} else {

enemy\_info.text += "\nRobust ("+string(Forcebuff[1])+"): Increases force spells damage and discard resistance by "+string(round((Forcebuff[0] - 1)\*100))+"%";

}

}

if DMGbuff[1] > 0 {

enemy\_info.text += "\nFervent ("+string(DMGbuff[1])+"): Spells deal "+string(round((DMGbuff[0] - 1)\*100))+"% more damage";

if global.equip[2] == 1 {

enemy\_info.text += ", (\*)Reflects "+string(round((DMGbuff[0] - 1)\*100)\*2)+"% damage back to attacker"

}

}

if meditatebuff > 0 {

enemy\_info.text += "\nFocused: Next spell has 50% empowerment (Misc. moves are excluded)";

}

enemy\_info.text += "\n\n(Click anywhere to close)";

}

**Событие Key Press – Control:**

//Переключение между ускоренным и нормальным режимом игры

if spedup.image\_alpha == 0 and global.active == noone and !instance\_exists(transition) {

game\_set\_speed(240,gamespeed\_fps);

spedup.image\_alpha = 1;

}

else {

game\_set\_speed(60,gamespeed\_fps);

spedup.image\_alpha = 0;

}

**Событие Key Press – Space:**

//Постановка и снятие игры с паузы

if global.timeflow {

global.timeflow = false;

paused = true;

pause.image\_alpha = 1;

}

else {

if paused {

global.timeflow = true;

paused = false;

pause.image\_alpha = 0;

}

}

**Событие Key Press – Q:**

//Переключение на режим разработчика и обратно

global.debug \*= -1

ЗАКЛЮЧЕНИЕ

В ходе данной курсовой работы были разработаны алгоритмы, закодирован и отложен программный продукт (интерактивная компьютерная игра) в среде программирования Game Maker Studio 2.

В ходе разработки был проведён анализ требований к программному продукту. С помощью созданной системы тестирования продукт был подтверждён на соответсвие заданных требований.

В ходе создания продукта была использована система контроля версий GitHub и создан дистрибутив для приложения.