МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»

Основная профессиональная образовательная программа

Направление подготовки 09.03.01 Информатика и вычислительная техника

Направленность (профиль) «Технологии разработки программного обеспечения»

форма обучения очная

Курсовая работа

«Организация и управление проектом по разработке компьютерной игры»

Обучающегося 3 курса

Цирулика Ивана Александровича

Научный руководитель:

Кандидат физико-математических наук, ассистент

Жуков Николай Николаевич

Санкт-Петербург

2019

–

Оглавление

ВВЕДЕНИЕ

Актуальность темы исследования: Актуальность обусловлена тем, что в настоящее время видеоигровая индустрия, сочетающая в себе огромное количество стеков технологий (см. приложение А), ставящая сложные задачи и требующая очень узкоспециализированные знания, является одной из самых передовых в IT и умение управлять в ней – является очень важным.

Объект исследования: Создание продуктов видеоигровой индустрии.

Предмет исследования: Управление процессом разработки видеоигры.

Цель исследования: Анализ процесса разработки видеоигры с применением agile-методологии.

Для достижения указанной цели в процессе выполнения работы были сформулированы следующие задачи:

1. Исследовать и проанализировать существующие agile-подходов
2. Разработать техническое задание продукта
3. Визуализировать план разработки при помощи диаграммы Ганта
4. Визуализировать процесс разработки при помощи диаграммы последовательностей
5. Применить инструменты управления проектом к процессу разработки видеоигр (Trello)

6. Провести unit-тестирование приоритетных модулей продукта

Глава 1 Теоретические основы управления разработкой видеоигры

1.1 Анализ agile-методологий и выбор наиболее подходящей

На данный момент в IT-индустрии преобладают гибкие методологии управления такие как SCRUM и Kanban. С целью выбора наиболее подходящего подхода, проведем сравнительный анализ.

В описаниях этих методик немало общего. Но на практике отличий предостаточно, для большей наглядности составим таблицу.

Scrum	**Kanban**
Продолжительность итераций/спринтов строго определена. Как правило, от двух недель до месяца.	Определяется именно продолжительность циклов.
Команда оценивает или планирует каждый спринт, исходя из информации в бэклоге.	Отслеживается рабочий процесс/рабочий элемент/Kanban-карта
В этой методике участникам отводятся три роли: владелец продукта, скрам-мастер и команда разработки.	Процесс построен без ролей.
После начала спринта изменения не допускаются.	В этом отношении Kanban более гибкая методика. Изменения вносятся в любое время.
Вся работа разбита на несколько этапов/спринтов.	Ход работы идет одним потоком.

На основе данной таблицы было принято решение воспользоваться методологией «SCRUM», так как она обеспечивает максимальную гибкость в разработке, что очень важно при темпах развития технологий CG (см. приложение А), являющимися основными в игровой индустрии, а так же постоянный контакт команд, занимающихся разработкой.

Глава 2 Управление процессом разработки

2.1 Выбор инструмента управления проектом

Для реализации подхода SCRUM был использован онлайн-сервис Trello. С его помощью аналоговый подход, с наклеиванием стикеров на стену и записи на них задач, заменяется цифровым аналоговым, который лучше сразу по нескольким пунктам: стена, отведенная под задачи, остается чистой, есть доступ даже у сотрудников, работающих удаленно и исключена потеря этих стикеров с задачами. Trello, даже в базовой версии, предоставляет широкий спектр возможностей для контроля проектов: создание досок (далее бордов) показывающих состояние проекта и состоящих из колонок, разделенных по определенному признаку, с задачами, а также возможность маркировать задания различными цветами. С целью упрощения работы над проектом было выделено 7 колонок:

- backlog — в ней находятся все задачи по проекту.
- to Do — задачи на спринт, которые переносятся из общей колонки (спринт — одна рабочая неделя).
- in Progress — задачи, к которым приступил исполнитель.
- on Review — просмотр имплементированных задач.
- committed — анализ и отработка комментариев.
- testing — задачи, которые тестируются.
- done — выполненные задачи.

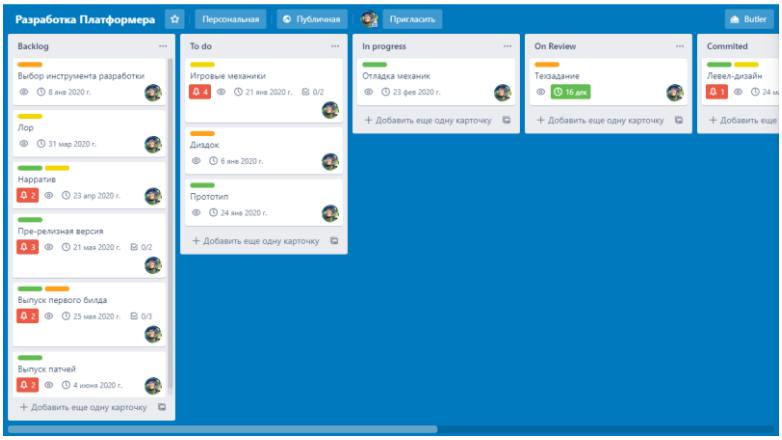

Рисунок 1 Доска Trello

2.2 Диаграмма Ганта

Для визуализации примерного плана разработки проекта, воспользуемся диаграммой Ганта. Диаграмма Ганта — это популярный тип столбчатых диаграмм (гистограмм), который используется для иллюстрации плана, графика работ по какому-либо проекту. Является одним из методов планирования проектов. Для ее построения воспользуемся программой Microsoft Excel. Для начала создадим таблицу – представляющую собой примерный план работы.

Название этапа	Начало	Длительность	Задержка	Конец
Выбор жанра видеоигры	02.12.2019	1	0	02.12.2019
Формулирование ТЗ	03.12.2019	14	0	16.12.2019
Создание игровых механик	17.12.2019	20	0	05.01.2020
Создание дизайн-доумента	04.01.2020	3	-2	06.01.2020
Выбор инструментов разработки	07.01.2020	2	0	08.01.2020
Имплементация игровых механик	08.01.2020	14	-1	21.01.2020
Создание первого прототипа игры для тестирования механик	15.01.2020	10	-7	24.01.2020
Отладка механик	25.01.2020	30	0	23.02.2020
Работа над левел дизайном	24.02.2020	30	0	24.03.2020
Создание лора игры	11.03.2020	21	-14	31.03.2020
Проработка игрового нарратива	25.03.2020	30	-7	23.04.2020
Создание пре-релизной версии	24.04.2020	14	0	07.05.2020
Тестирование	08.05.2020	14	0	21.05.2020
Выпуск первого билда игры	19.05.2020	7	-3	25.05.2020
Выпуск патчей	26.05.2020	10	0	04.06.2020

Рисунок 2 План проекта

Получившаяся диаграмма (см. Приложение В) наглядной показывает временные ресурсы, планируемые потратить на различные этапы разработки.

2.3 Техническое задание приложения

2.3.1 Назначение разработки

Приложение «Super Romario Bruh» является комплексным проектом, охватывающим различные аспекты разработки ПО;

содержит все основные аспекты компьютерной ролевой игры;

является продуктом сферы компьютерных развлечений.

2.3.2. Требования к программе или программному изделию

2.3.2.1 Требования к функциональным характеристикам

Данный проект является компьютерной игрой, вследствие чего предусматривается одна категория пользователей - игроки. В процессе работы приложения пользователь является непосредственным участником игрового процесса и оказывает непосредственное влияние на него.

Программа должна обладать следующим функционалом:

1. Графический функционал:
    a) создание окна;
    b) отрисовка графических примитивов;
    c) отрисовка текстур;
2. Звуковой функционал:
    a) регулировка общей громкости;
    b) регулировка громкости музыки;
    c) регулировка громкости внутри игровых звуков;
3. Внутри игровой функционал:
    a) система симуляции физики:
        i. Процедуры обработки столкновений;

ii. Процедуры симуляции физических явлений

b) система процедурной генерации карт;
c) боевая система;
d) система генерации противников
e) искусственный интеллект противников;
f) система генерации внутри игровых бонусов;
g) система регулировки игрового баланса;
h) система логирования;

4. Интерфейс пользователя:
   a) переходные сцены (вступительная, финальная, экран загрузки);
   b) главное меню;
   c) графический интерфейс пользователя.
      i. Шкала здоровья;
      ii. Шкала выносливости
      iii. Индикатор области карты

2.3.2.2 Требования к входным и выходным данным

Входными данными в компьютерной игре являются игровые настройки пользователя (разрешение экрана, громкость звуков, качество графики и т.д.), а также непосредственное управление во время игрового процесса с помощью компьютерной мыши и клавиатуры. Проект относится к играм в реальном времени, где в отличие от пошаговых действия игрока незамедлительно оказывают влияние на игровой процесс.

Выходными данными являются графическая интерпретация игрового процесса на мониторе игрока и звук, сопровождающий его. Действия игрока влияют на игровой процесс и текущее состояние игровой сцены. Игрок контролирует игрового персонажа с помощью интерфейса пользователя.

2.3.2.3. Требования к надежности

В программе должна присутствовать проверка входной информации на соответствие типов, принадлежность диапазону допустимых значений и соответствие структурной корректности. В случае возникновения ошибок предусмотреть возможность вывода информативных диагностических сообщений. В программе реализована система логов, а также возможность анализа дампа приложения в случае некорректного завершения.

2.3.2.4. Требования к составу и параметрам технических средств

Минимальные системные требования:

- ОС (операционная система): Windows XP/Vista/7/8;
  - Процессор: Intel Core 2 Duo @ 3.0 Ghz / AMD Athlon 64 X2 6000+;
- Оперативная память: 1 Gb;
- Жесткий диск: 10 Gb свободно;
- Видео память: 512 Mb;
- Видео карта: nVidia GeForce 9800 / AMD Radeon HD 4870;
- Звуковая карта: Совместимая с DirectX;
- DirectX 9.0c;
- Клавиатура, Мышь.

Рекомендуемые системные требования:

- ОС (операционная система): Windows Vista/7/8;
- Процессор: Intel Core i5 @ 3.2 GHz / AMD Phenom II X4 @ 3.6 GHz;
- Оперативная память: 2 Gb;
- Жесткий диск: 10 Gb свободно;
- Видео память: 1 Gb;
- Видео карта: nVidia GeForce GTX 460 / AMD Radeon HD 5850;

- Звуковая карта: Совместимая с DirectX;
- DirectX 11;
- Клавиатура, Мышь.

2.3.2.5. Требования к информационной и программной совместимости

Программа должна функционировать под управлением ОС семейства Windows следующих версий: Windows XP, Vista, 7, 8. В приложении используются библиотеки платформы.NET Framework. Также требуется установленный DirectX 9.0c или более поздней версии.

2.3.2.6. Требования к программной документации

Программная документация должна быть представлена руководством пользователя.

2.4 Макет приложения

Целью макета является визуальное схематическое изображение вида приложения. Он необходим в первую очередь для дизайн-документа и создается игровым дизайнером совместно с дизайнером уровней. Макет позволяет наглядно увидеть интерфейс будущей программы, и на основе обратной связи, полученной от заказчика, сделать правки. Макет четко показывает топологию графических объектов в окне приложения.

Пример макета (в 4 основных состояниях программы) представлен на изображениях ниже:

Рисунок 3 Макет главного меню видеоигры

Рисунок 4 Макет экрана загрузки уровня

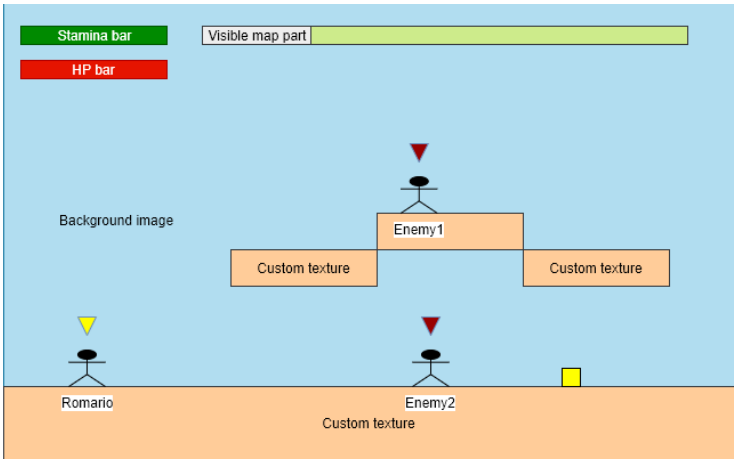

Рисунок 5 Макет игровой локации

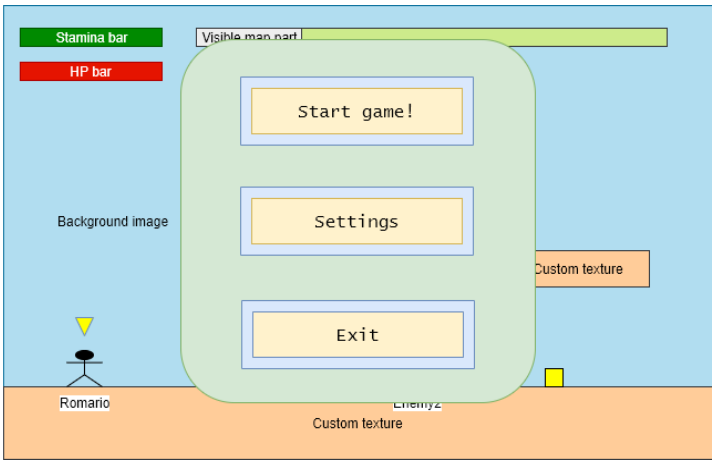

Рисунок 6 Макет вызова меню из игрового процесса

2.5 Диаграмма Состояний

Диаграмма состояний — это, по существу, диаграмма состояний из теории автоматов со стандартизированными условными обозначениями, которая может определять множество систем от компьютерных программ до бизнес-процессов. Используются следующие условные обозначения:

- Круг, обозначающий начальное состояние.
- Окружность с маленьким кругом внутри, обозначающая конечное состояние (если есть).
- Скруглённый прямоугольник, обозначающий состояние. Верхушка прямоугольника содержит название состояния. В середине может быть горизонтальная линия, под которой записываются активности, происходящие в данном состоянии.
- Стрелка, обозначающая переход. Название события (если есть), вызывающего переход, отмечается рядом со стрелкой.

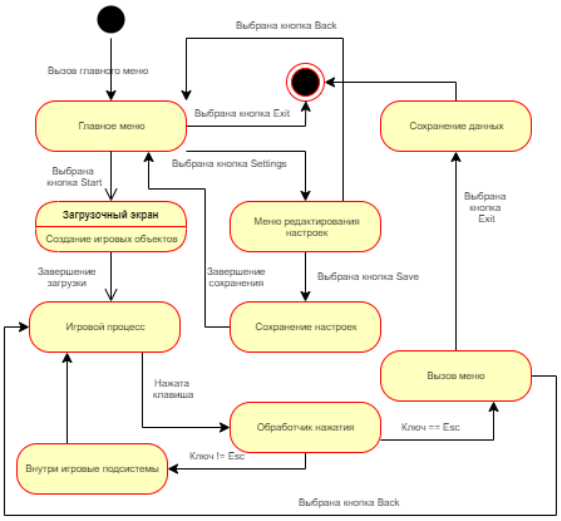

Рисунок 7 Диаграмма состояний

Unit-тестирование

Тестируемые функции

```
def moveLeft(self, k=1):
    self.coords[0] -= k
    return self.coords

def moveBot(self, k=1):
    self.coords[1] += k
    return self.coords

def moveRight(self, k=1):
    self.coords[0] += k
    return self.coords

def jump(self, k=1):
    self.coords[1] -= k
    return self.coords
```

Результаты тестирования

Рисунок 8 Результаты unit-тестирования

Код unit-теста смотреть в Приложении Б

ЗАКЛЮЧЕНИЕ

В процессе выполнения работы были решены следующие задачи:

1. Исследованы и проанализированы существующие agile-подходы
2. Разработано техническое задание продукта
3. Визуализирован план разработки при помощи диаграммы Ганта
4. Визуализирован процесс разработки при помощи диаграммы последовательностей
5. Применены инструменты управления проектом к процессу разработки видеоигр (Trello)
6. Проведено unit-тестирование приоритетных модулей продукта

Вывод: Разработка видеоигры сложный процесс, очень быстроменяющийся стек технологий CG и постоянный рост аппаратных требований к ним, требуют очень гибкой системы управления, так как инструментарий меняется буквально раз в месяц, одновременно с этим над разработкой трудится несколько команд, которые без связи друг другом не смогут адекватно функционировать, в связи с этим подход «SCRUM» показал свою эффективность. Его гибкость позволяет менять задачи даже раз в пару недель, а постоянный контакт команд, способствует хорошей коммуникации между ними, и благодаря этому процесс разработки продвигается наиболее быстро. Использование сервиса Trello позволяет эффективно использовать эту методологию, переназначая ответственных за ту или иную задачу, наличие таймера тоже дает свои плюсы в виде четких сроков сдачи для команд.

ЛИТЕРАТУРА

1. Allen Downey – ThinkPython+Kart[Python_3.2]
2. Briggs J. R. – Python for Kids – 2012
3. Davide Spallazzo, – Ilaria Mariani Location-Based Mobile Games. Design Perspectives – PoliMI SpringerBriefs, 2018.
4. Deitel H.M. et al. Python — How to Program
5. Gregory Jason - Game Engine Architecture. - A K Peters, Ltd, 2009. - 864
6. Вабищевич П. Н. Численные методы. Вычислительный практикум. – – 320 с.
7. Градов В.М. Компьютерное моделирование: Учебник / В.М. Градов, Г.В. Овечкин, П.В. Овечкин и др. - М.: Инфра-М, 2016. - 784 с.
8. Пилгрим Марк. Погружение в Python 3 (Dive into Python 3 на русском)
9. Прохоренок Н., Дронов В. Python 3 и PyQt 5. Разработка приложений. - СПБ: БХВ-Петербург, 2016. - 832 с.
10. Прохоренок Н.А. Python. Самое необходимое. — СПб.: БХВ-Петербург, 2011. — 416 с.
11. Хахаев И.А. Практикум по алгоритмизации и программированию на Python. – М.: Альт Линукс, 2010. — 126 с. (Библиотека ALT Linux).
12. Чаплыгин А.Н. Учимся программировать вместе с питоном.
13. Шапошникова С. Основы программирования на Python. Вводный курс.
14. Шрейер Джейсон Кровь, пот и пиксели. Обратная сторона индустрии. - СПБ.: Бомбора, 2019.

15. Scrum vs Kanban: в чем разница и что выбрать? // URL: https://habr.com/ru/company/hygger/blog/351048/ (дата обращения: 20.12.2019).

16. Синтез асинхронных конечных автоматов [Текст] / В. Г. Лазарев ; АН СССР. Институт проблем передачи информации. - М. : Наука, 1964. - 259 с. : ил. - Библиогр.: с. 252-256. - Б. ц.

ПРИЛОЖЕНИЯ

Приложение А

Глоссарий:

CG - область деятельности, в которой компьютеры наряду со специальным программным обеспечением используются в качестве инструмента как для создания (синтеза) и редактирования изображений, так и для оцифровки визуальной информации, полученной из реального мира, с целью дальнейшей её обработки и хранения.

Стек технологий - это набор технологий на основе которых ведется разработка.

Приложение Б

```
try:
    assert(player.coords == [0, 0])
    print(player.coords)
    assert(player.moveRight() == [1, 0])
    print(player.coords)
    assert (player.moveLeft() == [0, 0])
    print(player.coords)
    assert(player.moveRight(3) == [3, 0])
    print(player.coords)
    assert(player.moveLeft(2) == [1, 0])
    print(player.coords)
    assert(player.moveBot() == [1, 1])
    print(player.coords)
    assert(player.jump(2) == [1, -1])
    print(player.coords)
    print('Test succesfull')
except AssertionError as err:
    print('Test failed' + str(err))
```

Приложение В