КУРСОВЫЕ

№ п/п	Наименование дисциплин (модуля)	Тема курсовой работы (проекта)
1	Информатика	Арифметические основы ЭВМ
2	Комп. граф.	Разработка компьютерной игры "Побег из лаборатории"
3.	Теория автоматов	Синтез микропрограммного управляющего автомата
4.	РПС	Разработка приложения "Мобильный ВятГУ"
5	Разработка модулей СПО	Разработка программы поискового вебсервера методом движения муравьиных

Шестун Д. С.
(Фамилия, И.О. студента)

РАБОТЫ (ПРОЕКТЫ)

Семестр	Оценка	Дата сдачи	Подпись преподавателя	Фамилия преподавателя
II	отлично	05.06.15		Шихов
3	обычно	17.12.15		Кокошин
4	Хорошо	30.06.16		Мельцов
5	Хорошо	20.01.17		Чистяков
6	Хорошо	14.06.17		Шихов

Декан _____ (подпись)

ПРОВЕРЕНО

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Вятский государственный университет»
(«ВятГУ»)
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

Допущено к защите
Руководитель проекта
_____/Клюкин В. Л./
(подпись) (Ф.И.О)
«__»_____2015г.

РАЗРАБОТКА КОМПЬЮТЕРНОЙ ИГРЫ «ПОБЕГ ИЗ ЛАБИРИНТА»

Пояснительная записка
Курсовая работа по дисциплине
«Компьютерная графика»
ТПЖА 09.03.01.024 ПЗ

Разработал студент группы ИВТ-21 _____/Щесняк Д.С./

Руководитель доцент кафедры ЭВМ _____/Клюкин В.Л./

Проект защищен с оценкой «_____» _____
(оценка) (дата)

Члены комиссии _____/ _____/
(подпись) (Ф.И.О)

_____ /_____/

Киров 2015

Содержание

1 Введение

В наше время мы все чаще сталкиваемся с электронными вычислительными машинами, они буквально окружают нас, они используются во всех отраслях. Персональные компьютеры, ноутбуки, планшеты, телефоны, все это все сильнее входит в нашу жизнь. В 21 – м веке очень важно уметь владеть вычислительном машиной не на уровне пользователя, а на уровне программиста, который может создать любое программное обеспечение.

Все выше и выше поднимается уровень языков программирования. В настоящее время для того, чтобы уметь программировать не обязательно знать о тонкостях работы процессора, оперативной памяти и другого аппаратного обеспечения ЭВМ, достаточно уметь создавать алгоритмы и знать синтаксис языка.

Одним из примеров языка высокого уровня является python. Синтаксис языка очень минималистичен, но это не мешает ему пользоваться популярностью в очень разных областях разработки программного обеспечения.

Существует множество способов изучения языков программирования, по моему мнению, наиболее эффективным методом является метод интерактивного обучения: когда пользователь изучает язык программирования и параллельно с написание кода программы видит результат своей работы. Для обучения базовому синтаксису языка, отлично подходит игровая форма, т. к. она может наглядно продемонстрировать как именно работает программа. Поэтому в качестве программного обеспечения для обучения языку программирования python было выбрано именно игровое приложение.

2 Анализ аналогов

2.1 CodeCombat

Красочная игра с интересным и захватывающим сюжетом. Вам предстоит стать волшебником, который с помощью js-кода меняет окружающий мир. Для игры требуется обязательное подключение к интернету, по причине того, что она браузерная. Имеется поддержка мультиплеера. Поддерживает такие скриптовые языки как javaScript, Python, Lua и экспериментальные языки как CoffeScript. Скриншот игры приведен на рисунке 1. Данная игра находится по адресу http://codecombat.com/

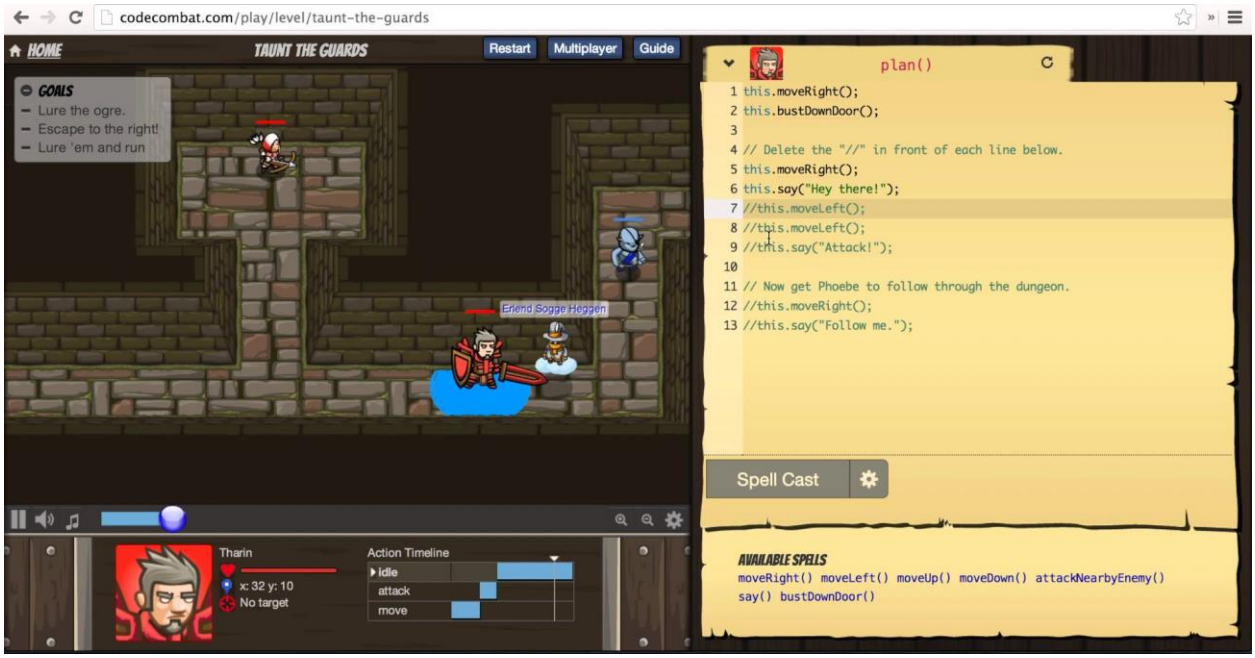


Рисунок 1 – Окно игры CodeCombat

2.2 CodeMonkey

CodeMonkey — это простая онлайновая игра, в процессе прохождения которой можно познакомиться с основами программирования. Весь процесс разделён на отдельные уровни. В любой момент можно прервать прохождение (прогресс сохраняется) или вернуться на несколько шагов назад. Программирование происходит при помощи псевдоязыка. Обязательное интернет подключение, отсутствует мультиплеер. На рисунке 2 представлен скриншот из данной игры. Данная игра находится по адресу https://www.playcodemonkey.com/

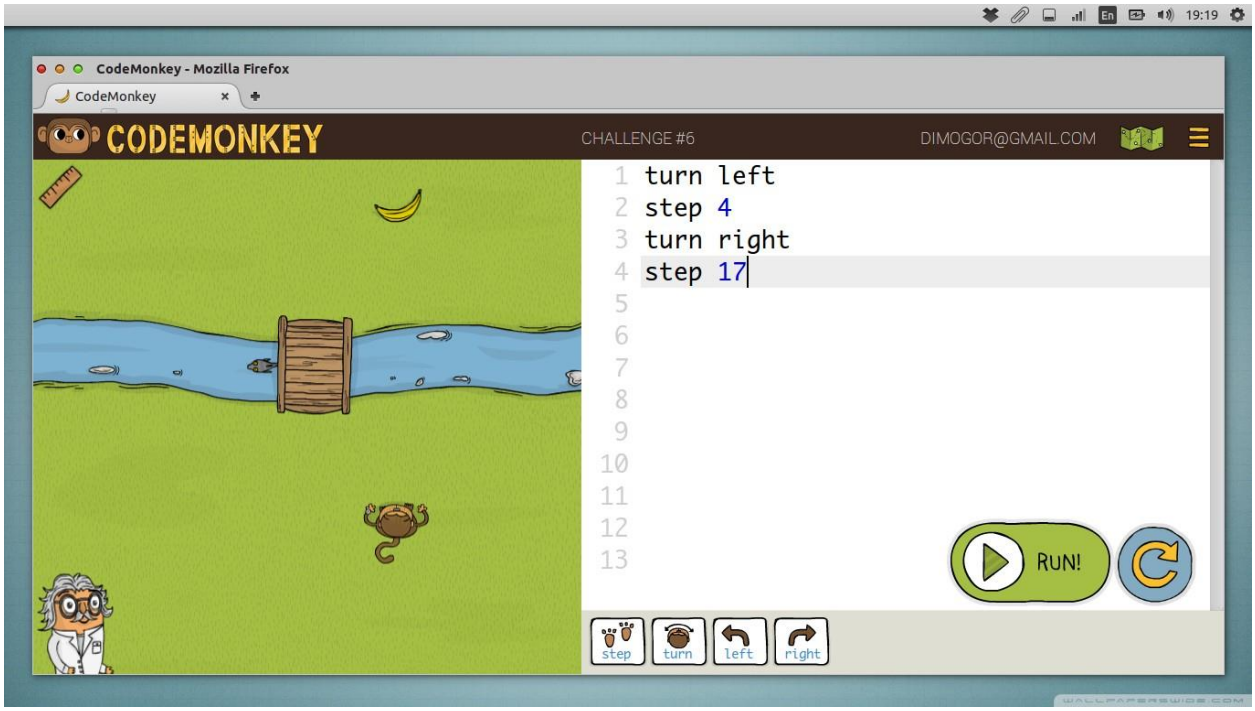


Рисунок 2 – окно игры CodeMonkey

2.3 Bit’s Quest

В игре нужно будет с помощью JavaScript-кода управлять ботом, выполняя задания к каждому уровню. Самое частое задание — добраться до выхода из лабиринта, но иногда приходится сражаться с вражеским кораблем.

Вам предстоит разобраться, что такое события, как использовать функции, и может понадобиться применить замыкания. Для игры требуется обязательное подключение к интернету. На рисунке 3 представлен скриншот окна игры. Данная игра находится по адресу http://bitsquest.bitbucket.org/

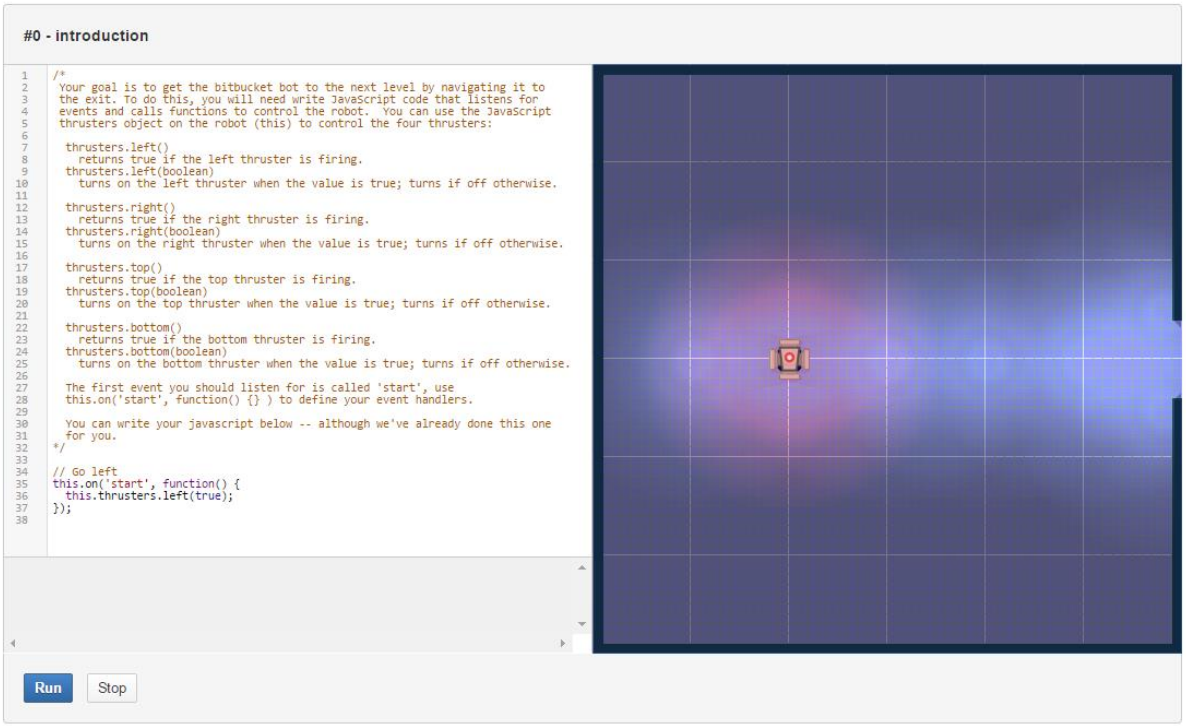


Рисунок 3 – главное окно игры Bit’s Quest

2.4 Ruby Warrior

Сценарий игры: руби-воин должен убить всех врагов. Управлять героем нужно с помощью Ruby-кода.

Для прохождения нужно разбираться в коде, но знаний по Ruby будет достаточно самых базовых. Вас не будут учить программировать, зато заставят решать логические задачи, причем иногда очень сложных. Главное окно программы представлено на рисунке 4. Данная игра находится по адресу https://www.bloc.io/ruby-warrior/

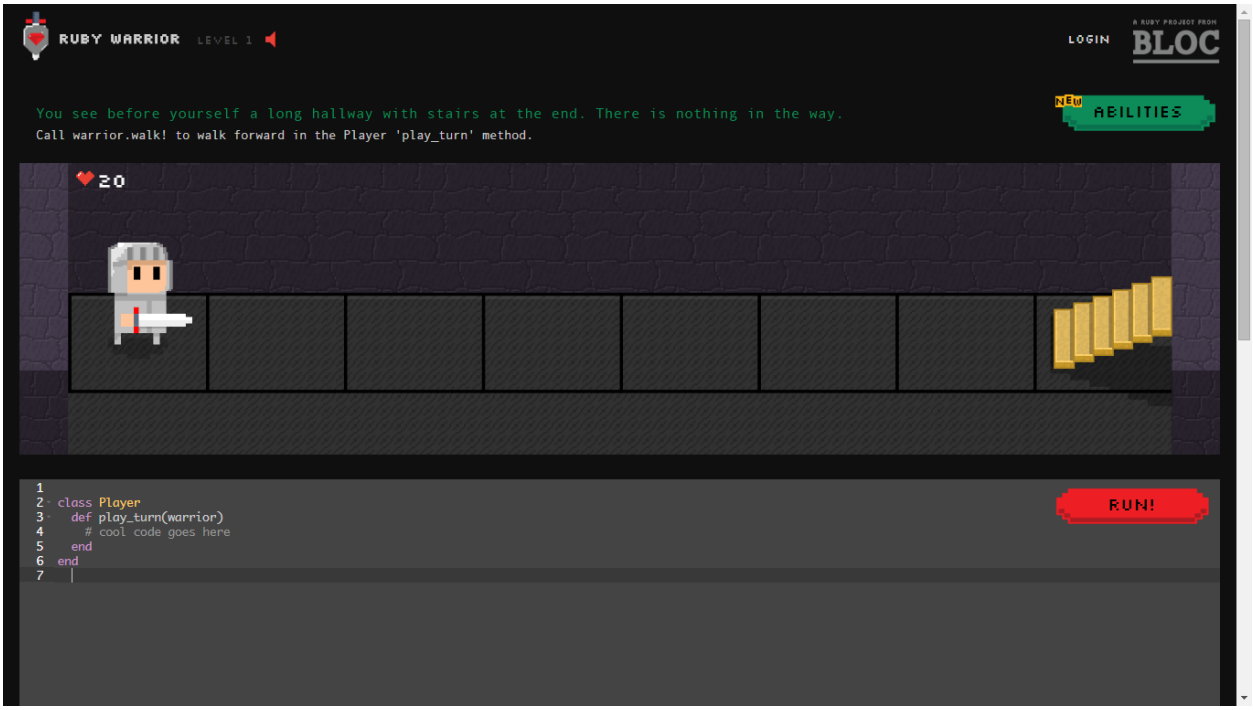


Рисунок 4 – главное окно программы Ruby Warrior

Таблица 1 сравнение аналогов

Критерий / Аналог	Низкий порог вхождения	Обязательное постоянное интернет соединение	Наличие многопользовательской игры	Обучение осуществляется на реально существующих языках
CodeCombat	+	+	+	+
CodeMonkey	+	+	-	-
Bit’s Quest	-	+	-	+
Ruby Warrior	-	+	-	+

3 Постановка задачи

На основе анализов аналогов можно установить критерии, которым должен удовлетворять конечный продукт для возможности конкурирования с представленными аналогами.

Выходное программное обеспечение должно:

- Иметь низкий порог вхождения пользователей.
- Использовать в качестве обучающего языка реальный язык программирования.
- Минимально зависеть от интернет соединения конечного пользователя.
- Иметь возможность многопользовательской игры, а именно сервер логических битв.

В качестве обучающего языка программирования был выбран Python. Python – язык программирования высокого уровня общего назначения, который ориентирован на уменьшение количества кода, сохраняя при этом его читаемость.

Плюсы python:

- Легкий для понимания синтаксис научит пользователя блочной модели программирования
- Огромное количество библиотек для расширения функционала
- Возможность перенести программу на любую платформу на которой есть интерпретатор.
- Благодаря нестрогой типизации программы на python создаются намного быстрее чем на C++/pascal.

Итог: Конечны продукт должен быть доступен как можно большему числу пользователей, другими словами, должен иметь низкий порог вхождения по знаниям в программировании, должен минимально зависеть от интернет – соединения пользователя и иметь возможность многопользовательской игры. В качестве обучающего языка использоваться интерпретатор языка python 3.

3.1 Требования к аппаратному обеспечению.

Процессор: Pentium 4/Athlon XP 3 ГГц

ОЗУ: 1 Гб памяти

Видеокарта: Nvidia Geforce 6600 и лучше

Жесткий диск: 10мб

3.2 Требования к программному обеспечению.

Операционная система: Windows XP и выше

Интерпретатор Python версии 3.0 и выше

3.3 Требования к интерфейсу

Интерфейс должен быть максимально простым и понятным конечному пользователю. Главное меню не должно содержать излишнее количество пунктов. Главное окно программы должно иметь графического окно, в котором будет визуализироваться весь код написанный пользователем, главное оно должно иметь форму ввода исполняемого кода, и форму вывода, в которой будут выводится ошибки.

4 Схема взаимодействия функциональных блоков.

Схема взаимодействия функциональных блоков представлена на рисунке 5

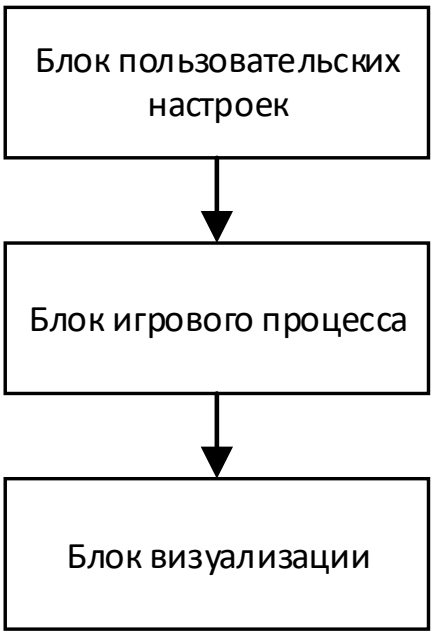


Рисунок 5 – схема взаимодействия функциональных блоков

4.1 Блок пользовательских настроек

Блок пользовательских настроек включается в себя функции, которые загружают, сохраняют и обрабатывают данные пользователя. А именно: его имя, уровень, который он проходит и статистика по написанному коду.

4.2 Блок игрового процесса

Блок игрового процесса содержит в себе функции обработки нажатий кнопок, загрузки карты, исполнения на интерпретаторе кода : пользователь вводит в специальное поле свои исходный код программы, нажимает на кнопку исполнения, программа передается интерпретатору python, который исполняет ее и заносит в лог всю информацию. Также этот блок включает в себя функцию проверки на прохождение пользователем уровня: если персонаж находится на клетке выхода, то пользователю засчитывается прохождение уровня, иначе нет.

4.3 Блок визуализации

Блок визуализации содержит в себе функции отвечающие за прорисовку на экран: функция рисования карты, анимация движения главного героя, своевременная перерисовка экрана.

5 Математический аппарат

Для графического изображения общего принципа работы программы был выбран ориентированный граф, который представлен на рисунке 6.

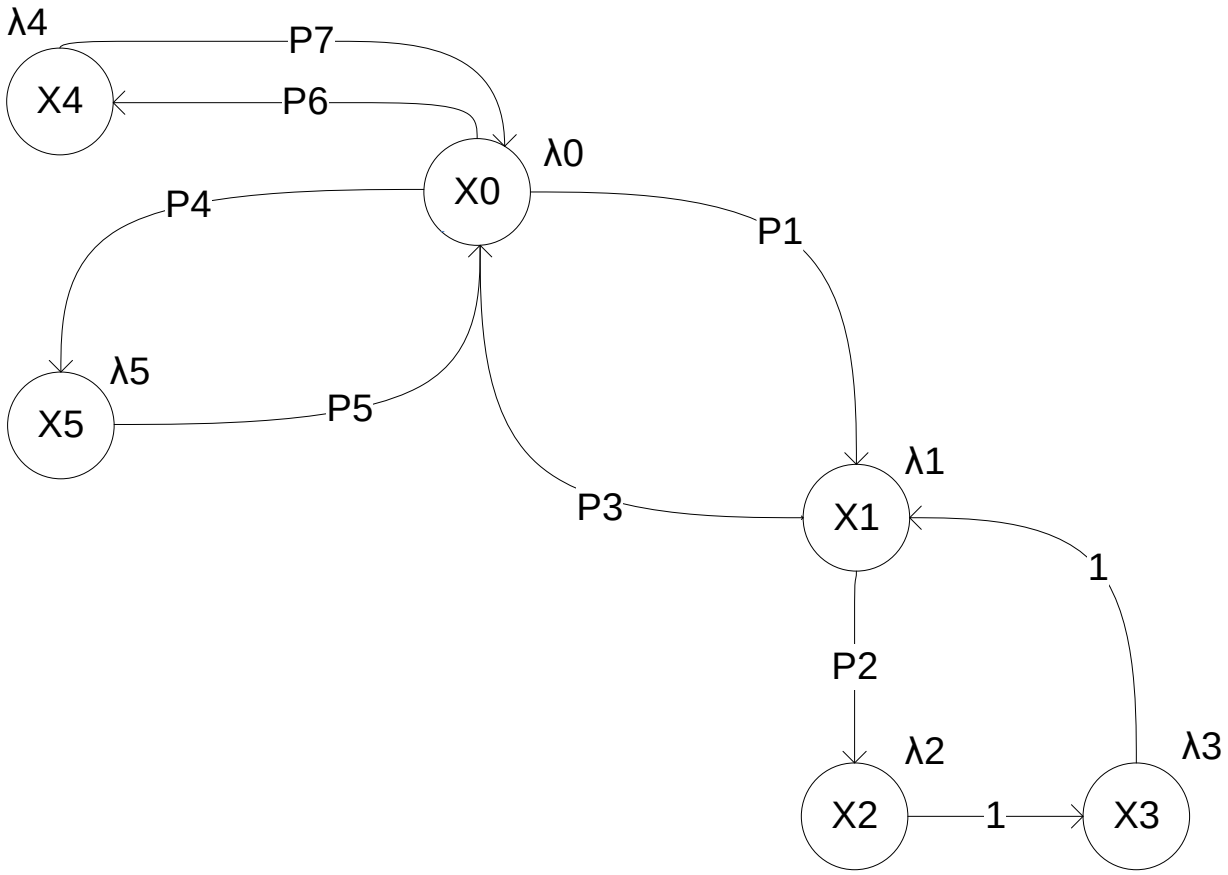


Рисунок 6 – общая схема работы программы

X0 - Главное меню
X1 - Игровой процесс
X2 - Исполнение кода пользователя
X3 - Визуализация результата
X4 - Меню смены пользователя
X5 - Окно справки

Λ0 - Визуализация главного меню
λ1 - Визуализация игрового экрана
λ2 - Исполнение кода пользователя
λ3 - Визуализация алгоритма игрока
λ4 - Визуализация окна ввода пользователя
λ5 - Визуализация окна справки

P1 - Нажатие на кнопку "Начать игру"
P2 - Нажатие кнопки старт в окне игры
P3 - Нажатие кнопки возврата в окне игры
P4 - Нажатие кнопки справка
P5 - Нажатие на конпку закрыть в окне справки
P6 - Нажатие на кнопку сменить пользователя
P7 - Нажатие на кнопки ОК в окне смены пользователя

6 Разработка схем алгоритмов

Блок – схемы алгоритмов представлены в приложений А

7 Разработка программного обеспечения

7.1 Разработка модульной структуры

Игру было решено разбить на 5 модулей: модуль создания окна, модуль главного меню, модуль игры, модуль ядра, модуль тестовой системы.

Модуль создания окна представляет из себя алгоритмы прорисовки окна, перерасчета размера объектов, при изменении размеров окна, загрузка файла и сохранение файла конфигурации.

Модуль главного меню представляет из себя алгоритм построения главного меню, обработчики событий при взаимодействием с элементами главного меню

Модуль игры включает в себя визуализатор кода пользователя, отслеживание игровых событий.

Модуль ядра включает в себя все необходимые классовые структуры, а именно: класс фонового изображения, необходимы для визуализации фона главного меню, класс пунктов меню, класс текстовых сообщений, класс всплывающих сообщений.

Модуль тестовой системы включает в себя алгоритм исполнения исходного кода пользователя и последующего ведения лога.

Модульная структура представлена на рисунке 7.

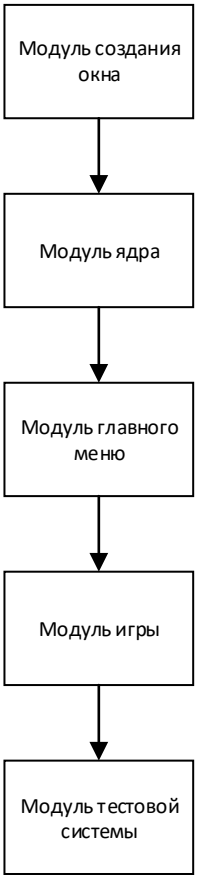


Рисунок 7 – модульная структура игры

7.2 Разработка пользовательского интерфейса

Следуя из требований, конечный продукт должен иметь наиболее простой и понятный пользователю интерфейс.

Главное меню будет состоять из:

- Начать игру
- Сменить профиль
- Справка
- Выход

Окно игры будет состоять из:

- Поле ввода исходного кода
- Кнопка исполнения исходного кода пользователя
- Холст на котором будет прорисовываться все действия пользователя

Экранные формы пользовательского интерфейса представлены в приложении В.

7.3 Исходный код

Исходный код программы представлен в приложении Б

8 Заключение

Результатом работы над курсовым проектом является 2-х мерная игра, обучающая языку программирования python. Конечная программа получилась согласно требованиям предъявленных к ней, а именно:

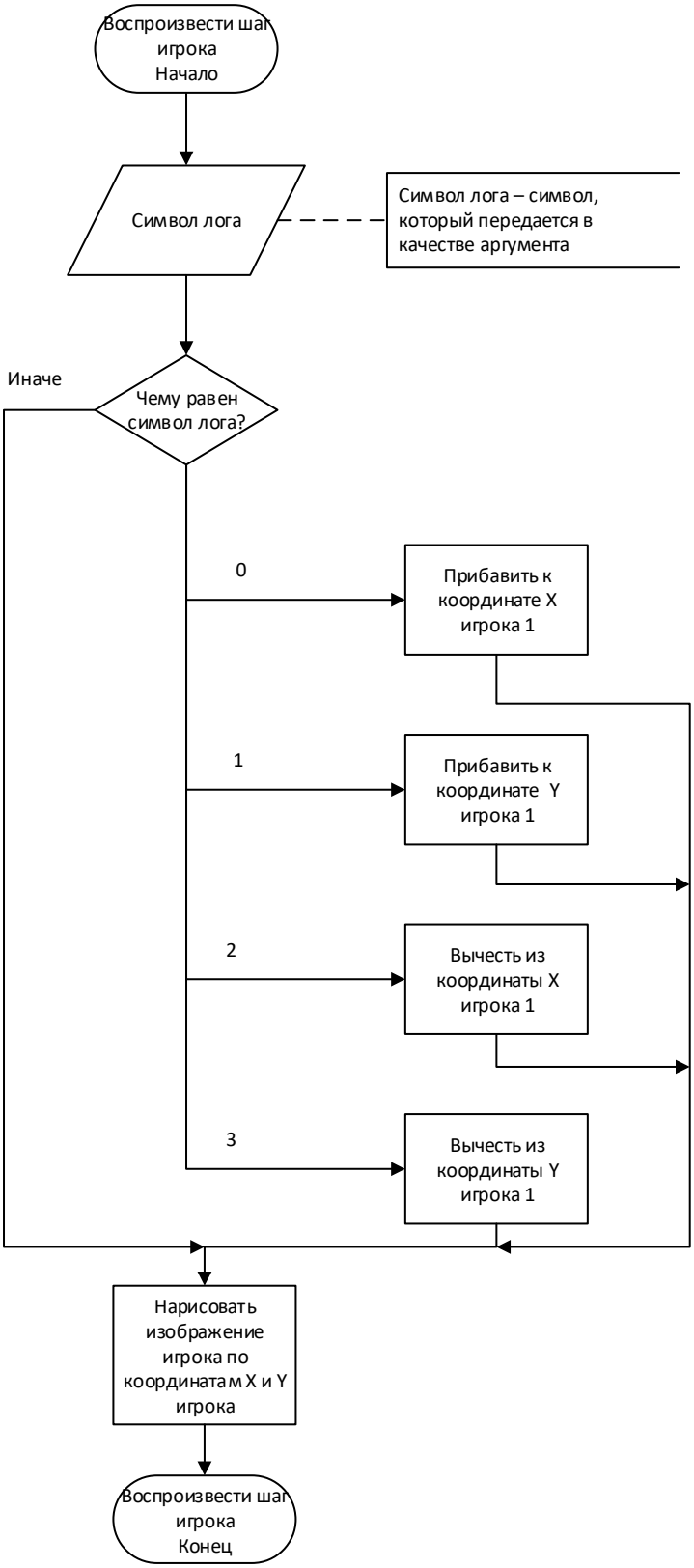
- Имеет низкий порог вхождения пользователей.
- В качестве обучающего языка используется реально существующий язык - python
- Для работы игры не требуется наличие интернет - соединения
- Разработан модуль, который осуществляет функцию битв логических стратегий

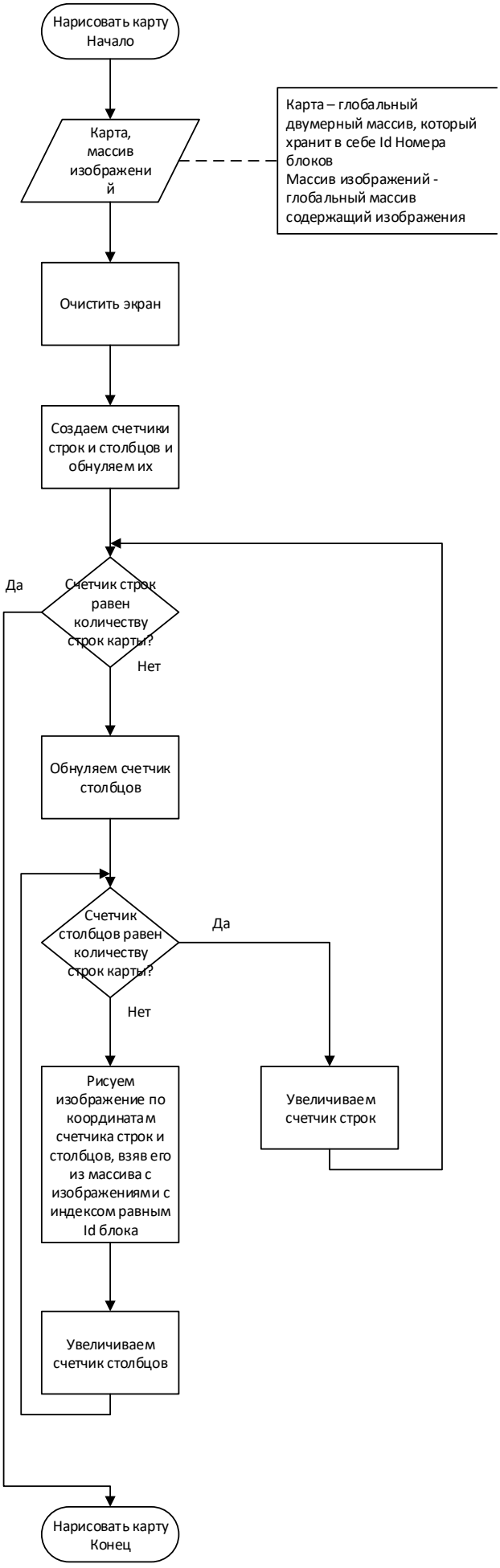
В дополнение к этому, вместе с основной игрой добавлен редактор карт для возможности создания собственных уровней, в том числе и для битв логических стратегий.

Главным преимуществом данной игры заключается в возможности игры в нее без наличия интернет – подключения.

Приложение А

Блок-схемы алгоритмов


Воспроизвести шаг игрока
Начало
Символ лога
Символ лога – символ, который передается в качестве аргумента
Иначе
Чему равен символ лога?
0
Прибавить к координате X игрока 1
1
Прибавить к координате Y игрока 1
2
Вычесть из координаты X игрока 1
3
Вычесть из координаты Y игрока 1
Нарисовать изображение игрока по координатам X и Y игрока
Воспроизвести шаг игрока
Конец


Нарисовать карту
Начало
Карта,
массив
изображени
й
Карта – глобальный двумерный массив, который хранит в себе Id Номера блоков
Массив изображений - глобальный массив содержащий изображения
Очистить экран
Создаем счетчики строк и столбцов и обнуляем их
Да
Счетчик строк равен количеству строк карты?
Нет
Обнуляем счетчик столбцов
Счетчик столбцов равен количеству строк карты?
Да
Нет
Рисуем изображение по координатам счетчика строк и столбцов, взяв его из массива с изображениями с индексом равным Id блока
Увеличиваем счетчик строк
Увеличиваем счетчик столбцов
Нарисовать карту
Конец

Приложение Б

Исходный код

Модуль Core.py

```
from tkinter import *
import os
import random
import math
from mainScreen import *

# CONSTS
SIZE = 64
DX = 450
DY = 96

# Вспомогательные функции
def absToRel(point):
    return [int((point[0] - DX)//SIZE), int((point[1] - DY)//SIZE)]

class Background(object):
    """background class"""
    def __init__(self, canv):
        super(Background, self).__init__()
        self.canv = canv
        self.isMoveRight = True
        self.dx = 1
        self.dy = 0
        self.imageList = list(filter(lambda x : x.endswith('.gif'), os.listdir('bg')))
        path = 'bg\\' + random.choice(self.imageList)
        self.img = PhotoImage(file = path).zoom(round(WIDTH/width), round(WIDTH/width))
        self.id = self.canv.create_image(-200, 0, image = self.img, anchor = 'nw', tag =
'background')
```

```
        def changeImage(self):
                path = 'bg\\' + random.choice(self.imageList)
                self.img = PhotoImage(file = path).zoom(round(WIDTH/width), round(WIDTH/width))
                self.canv.itemconfigure(self.id, image = self.img)
                pColor = self.img.get(1, 1)
                # self.canv['bg'] = '#' + str(pColor[0]) + str(pColor[1]) + str(pColor[2])



        def changeState(self):
                if self.canv.coords(self.id)[0] > 0 and self.isMoveRight:
                        self.isMoveRight = False
                        self.dx = -1
                        self.changeImage()
                elif self.canv.coords(self.id)[0] < -200 and not self.isMoveRight:
                        self.isMoveRight = True
                        self.dx = 1
                        self.changeImage()


        def mainloop(self):
                try:
                        self.changeState()
                        self.canv.move(self.id, self.dx, self.dy)
                        self.canv.after(50, self.mainloop)
                except:
                        pass


class MenuElement(object):
        """docstring for menuElement"""
        def __init__(self, canv, x, y, text):
                super(MenuElement, self).__init__()
```

```
                self.canv = canv
                self.x = x
                self.y = y
                self.color = 255
                self.leave = True
                self.id = self.canv.create_text(x, y, text = text, anchor = 'nw', font = 'Arial 15 italic bold',
fill = '#000000', tag = 'menuElement')
                # self.tag = 'menu' + self.id
                # self.canv.create_rectangle(x, y, x + 100, y + 20, fill = 'gray')
                self.canv.tag_bind(self.id, '<Motion>', lambda event: self.move(self.x + 40, 0))
                self.canv.tag_bind(self.id, '<Leave>', lambda event: self.moveBack())
                # self.canv.tag_bind(self.id, '<B1-Motion>', self.drag)

        def move(self, x, y):
                coords = self.canv.coords(self.id)
                sign = math.copysign(1, x - coords[0])
                self.canv.move(self.id, 1*sign, 0)
                if coords[0] < x:
                        self.canv.after(10, lambda : self.move(x, y))

        def moveBack(self):
                coords = self.canv.coords(self.id)
                sign = math.copysign(1, self.x - coords[0])
                self.canv.move(self.id, 1*sign, 0)
                if coords[0] != self.x:
                        self.canv.after(10, lambda : self.moveBack())

        def drag(self, event):
                self.canv.coords(self.id, event.x, event.y)

        def onClick(self, func):
                self.canv.tag_bind(self.id, '<Button-1>', func)
```

```
class PopupMessage(object):
        """docstring for PopupMessage"""
        def __init__(self, canv, x, y, width, height, message):
                super(PopupMessage, self).__init__()
                message = message.split('\n')

                self.x = x
                self.y = y
                self.width = width
                self.height = height
                self.canv = canv
                if width < (len(message[0]) * 13):
                        self.width = len(message[0]) * 13
                self.id = self.canv.create_rectangle(x - 3, y - 3, x + self.width + 3, y + height + 3, fill = "white", tag = "popupMessage")
                self.canv.create_rectangle(x, y, x + self.width, y + height, fill = "black", tag = "popupMessage" + str(self.id))

                for i, line in enumerate(message):
                        self.canv.create_text(x + 25, (y + 3) + 15 * i, text = line, anchor = 'nw', fill = "white", font = "Arial 13 italic bold", tag = ("popupMessage" + str(self.id), 'Text'))
                self.canv.itemconfig(self.id, tag = ("popupMessage" + str(self.id), "border"))

        def move(self, dx, dy):
                x = self.x + dx
                y = self.y + dy
                c = math.hypot(x - self.x, y - self.y)
                if c > 1:
                        self.x += round(dx / c)
                        self.y += round(dy / c)
                        dx -=  dx/c
```

```
                dy -= dy/c
                self.canv.move("popupMessage" + str(self.id), round((x - self.x) / c), round((y - self.y) / c))
                self.canv.after(40, lambda : self.move(dx, dy))

        def show(self, dx, dy, time):
            self.move(dx, dy)
            self.canv.after(time, lambda : self.move(-dx, -dy))

        def onClick(self, func):
            self.canv.tag_bind("popupMessage" + str(self.id), "<Button - 1>", lambda x: func(x));


    def showMessage(canv, x, y, width, height, message):
        message = message.split('\n')

        # canv.create_rectangle(400, 50, 1100, 650, fill="#fbd3a0", tag = 'showMessage')
        canv.create_rectangle(x - 3, y - 3, x + width + 3, y + height + 3, fill = "white", tag = 'showMessage')
        canv.create_rectangle(x, y, x + width, y + height, fill="#000000", tag = 'showMessage')
        # canv.create_rectangle(dx - borderWidth, dy - borderWidth, dx + 8*SIZE + borderWidth, dy + 8*SIZE + borderWidth, fill = '#3c2418', tag = 'settings')

        # Рисуем крестик
        # canv.create_rectangle(x + width - 32, y + 2, x + width - 3, y + 32, fill = '#fdf5c2', tag = 'closeBut showMessage')
        canv.create_line(x + width - 15, y + 5, x + width - 5, y + 15, width = 4, fill = "red", tag = 'closeBut showMessage')
        canv.create_line(x + width - 15, y + 15, x + width - 5, y + 5, width = 4, fill = "red", tag = 'closeBut showMessage')

        # canv.create_rectangle(x + width - 50, y + height - 30, x + width - 10, y + height - 5, fill = '#fdf5c2', tag = 'closeBut showMessage')
        # canv.create_text(x + width - 50, y + height - 30, text = 'OK', anchor = 'nw', font = 'Arial 15 italic bold', tag = 'closeBut showMessage')
```

```
                for i, j in enumerate(message):
                        canv.create_text(x + 25, y + 50 + 25*i, text = j, anchor = 'nw', font = 'Arial 15 italic bold',
fill = 'white', tag = 'showMessage')

                canv.tag_bind('closeBut', '<Button-1>', lambda x: canv.delete('showMessage'))
                canv.tag_bind('closeBut', '<B1-Motion>', lambda x: canv.itemconfig('closeBut', fill = '#FF0000'))

        class InputDialog(object):
                """docstring for InputDialog"""
                def __init__(self):
                        super(InputDialog, self).__init__()
                        self.top = Toplevel(mainScreen)
                        self.top.config(width = 100, height = 50)
                        Label(self.top, text = "Nickname: ").pack()
                        self.e = Entry(self.top)
                        self.e.pack(padx = 5)
                        self.e.config(selectborderwidth = 0)
                        self.okBut = Button(self.top, text = "OK", width = 20)
                        self.okBut.pack(pady = 5)

                def onClick(self, func):
                        self.okBut.config(command = (lambda: func(self)))
```

Модуль MainMenu.py

```
from mainScreen import *
from core import *
import os
import game

helpMessage = """
Добро пожаловать в игру Побег из лабиринта.

Главная цель этой игры - научить вас основам программирования.
Просто играите и наслаждайтесь.


Игру разработал студент ИВТ-21, Даниил Щесняк.
"""


def mainloop():
    global config

    canv = Canvas(mainScreen, bg="white")
    canv.pack(expand=YES, fill=BOTH)

    bg = Background(canv)
    startBut = MenuElement(canv, 50, 600, 'Начать игру')
    settingsBut = MenuElement(canv, 50, 625, 'Другой аккаунт')
    helpBut = MenuElement(canv, 50, 650, 'Справка')
    exitBut = MenuElement(canv, 50, 675, 'Выход')

    # loadConfig('Nellrun')
    # print(config)
```

```
        player = PopupMessage(
            canv, 500, -35, 200, 30, "Добро пожаловать, " + config["name"])

        player.show(0, 35, 5000)
        player.onClick(lambda event: InputDialog().onClick(lambda self: canv.destroy() or
loadConfig(self.e.get()) or mainloop() or self.top.destroy()))

        startBut.onClick(lambda event: canv.destroy() or game.mainloop())
        settingsBut.onClick(lambda event: InputDialog().onClick(lambda self: canv.destroy() or
loadConfig(self.e.get()) or mainloop() or self.top.destroy()))
        helpBut.onClick(
            lambda event: showMessage(canv, 400, 50, 700, 500, helpMessage))
        exitBut.onClick(lambda event: mainScreen.quit())

        # Запуски циклов жизни
        bg.mainloop()
```

Модуль Game.py

```
from mainScreen import *
from core import *
import mainMenu
from PIL import ImageTk, Image
import os
import threading


class PlayerAnim(object):
    """docstring for PlayerAnim"""
    def __init__(self, canv, loc):
        super(PlayerAnim, self).__init__()
        self.angle = 0
        self.x = 0
        self.y = 0
        self.x0 = 0
        self.y0 = 0
        self.step = 0
        self.canv = canv

        for y, l in enumerate(loc):
            for x, block in enumerate(l):
                if block == 2:
                    self.x = x
                    self.y = y
                    self.x0 = x
                    self.y0 = y
                    break

        self.sprite = []
        playerImg = Image.open('img//player48.png')
```

```
        imgs = []
        for i in range(0, 6):
            for j in range(0, 10):
                img = playerImg.crop((48 * j, 48 * i, 48 * (j + 1), 48 * (i + 1)))
                imgs.append(img)

        for i in range(0, 240):
            img = ImageTk.PhotoImage(imgs[i % 60].rotate(-90 * ( i // 60)))
            self.sprite.append(img)

        self.img = self.canv.create_image(self.x * 48, self.y * 48, image = self.sprite[0], anchor = 'nw')

    def playLogs(self, loc, logs, frame, n):

        angle = int(logs[0])
        STEP = 0.05

        self.canv.coords(self.img, self.x * 48, self.y * 48)
        self.canv.itemconfigure(self.img, image = self.sprite[60 * angle + frame])

        if angle == 0:
            self.x += STEP
        elif angle == 1:
            self.y += STEP
        elif angle == 2:
            self.x -= STEP
        else:
            self.y -= STEP

        if n < (1 / STEP) - 1:
            self.canv.after(15, lambda: self.playLogs(loc, logs, (frame + 1) % 60, n + 1))
```

```
            else:
                if len(logs) > 1:
                    self.playLogs(loc, logs[1::], (frame + 1) % 60, 0)
                else:
                    if loc[round(self.y)][round(self.x)] == 3:
                        config["level"] = str(int(config["level"]) + 1)
                        saveConfig()
                        canv.delete('all')
                        game()



def loadMap(name):
    loc = []
    if os.path.isfile('maps//' + name):
        mapFile = open('maps//' + name, 'r')
        for line in mapFile.readlines():
            loc.append([int(i) for i in line.rstrip().split(' ')])
    else:
        global canv
        canv.destroy()
        mainMenu.mainloop()
    return loc


def drawMap(canv, loc):
    for x in range(20):
        for y in range(20):
            canv.create_image(48 * x, 48 * y, image = textureList[1], anchor = 'nw', tag = 'block')


    for i, line in enumerate(loc):
        for j, block in enumerate(line):
```

```
            canv.create_image(48 * j, 48 * i, image = textureList[block], anchor = 'nw', tag = 'block')


def onStartButClick():
    global player
    global codeInput
    global canv

    open('tmp.py', 'w').write(codeInput.get(1.0, END))

    os.system('python -m testSystem.py tmp.py')
    logsFile = open('logs//' + config["level"] + '.maptmp.py.log')
    loc = loadMap(logsFile.readline().rstrip())
    logs = logsFile.readline().rstrip()
    errors = logsFile.readline().rstrip()
    logsFile.close()

    if (len(errors) > 0) and (errors != '3'):
        PopupMessage(canv, 0, -50, 0, 30, errors).show(0, 50, 5000)

    player.x = player.x0
    player.y = player.y0
    player.playLogs(loc, logs, 0, 0)

textureList = []
for i in range(0, 4):
    img = ImageTk.PhotoImage(file = 'img//' + str(i) + '.bmp')
    textureList.append(img)


def newThread(f):
```

```
    global t
    t = threading.Thread(target = f)
    return t


codeInput = 0
player = 0
canv = 0
t = threading.Thread()
butImg = ImageTk.PhotoImage(file = 'img/startBut.png')

def mainloop():
    global codeInput
    global player
    global canv

    canv = Canvas(mainScreen, bg="#1c60ab", width = 900)
    codeInput = Text(mainScreen, bg = "white", width = 400, fg = "#1c60ab", font = 'Arial 10 bold')
    startBut = Button(mainScreen, image = butImg, bg = '#EEEEEE', font = 'Arial 15 bold', command =
lambda : newThread(onStartButClick).start())
    startBut.config(relief = FLAT)

    canv.pack(expand=YES, fill=Y, side = LEFT)
    startBut.pack(side = BOTTOM, expand = NO)
    codeInput.pack(side = LEFT, expand = YES, fill = Y)
    game()

def game():
    global canv
    global player
    global codeInput
```

```
loc = loadMap(config['level'] + '.map')
codeInput.delete(1.0, END)
codeInput.insert(END, open('maps//' + config['level'] + '.py', 'r').read())
drawMap(canv, loc)


player = PlayerAnim(canv, loc)
```

Приложение В

Экранные формы

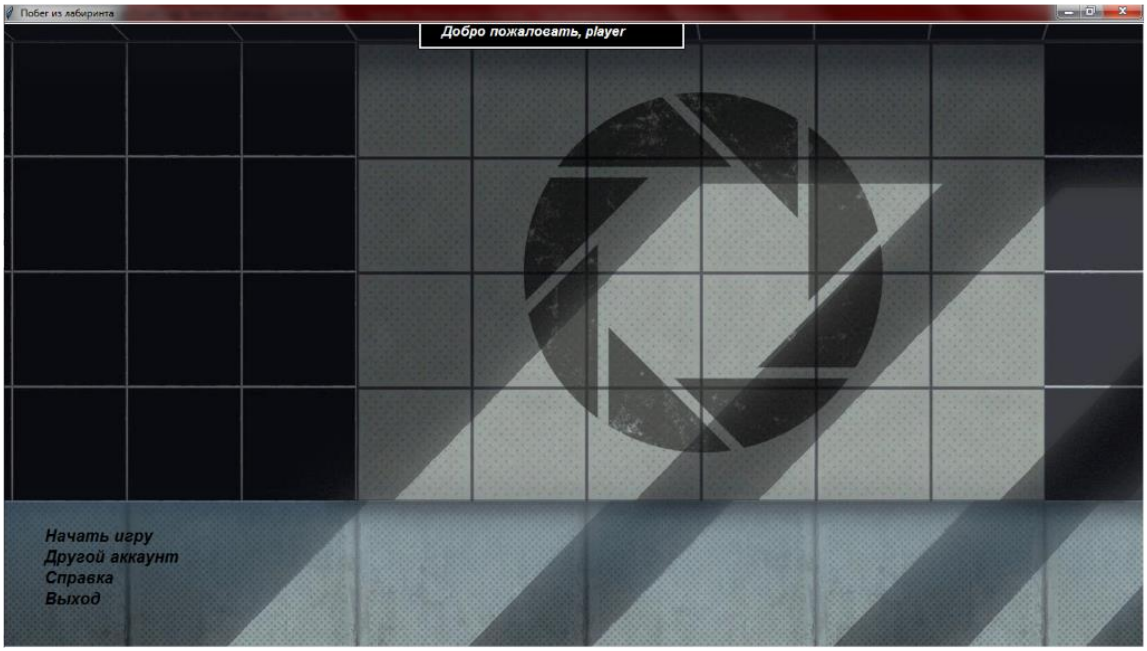


Рисунок 8 – Главное меню программы

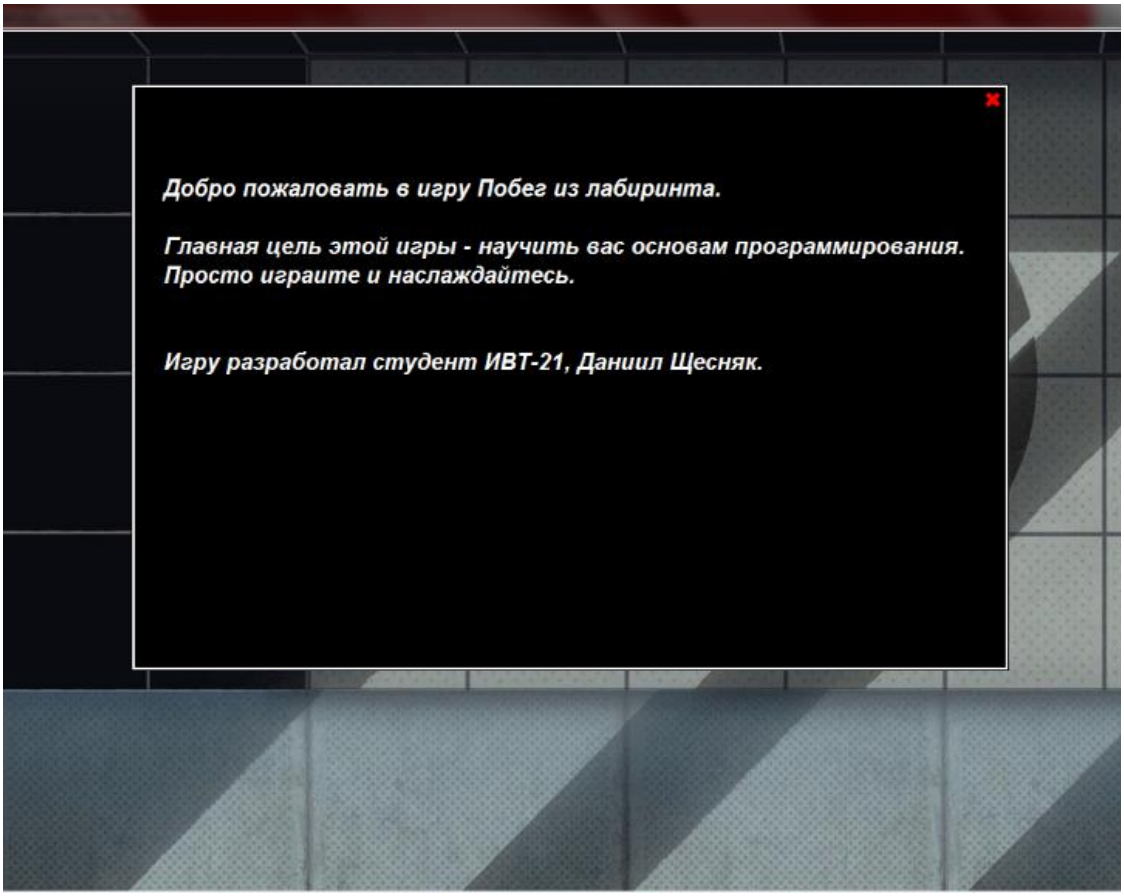


Рисунок 9 – Окно справки

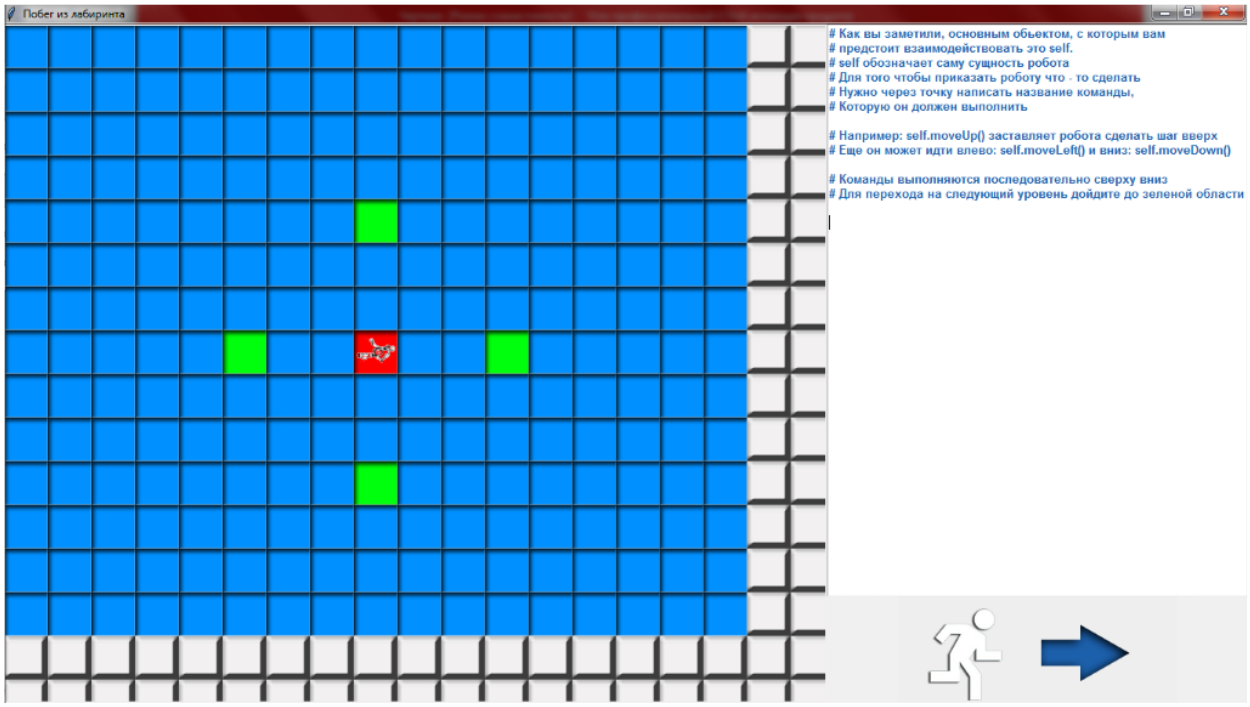


Рисунок 10 – Экран игрового процесса

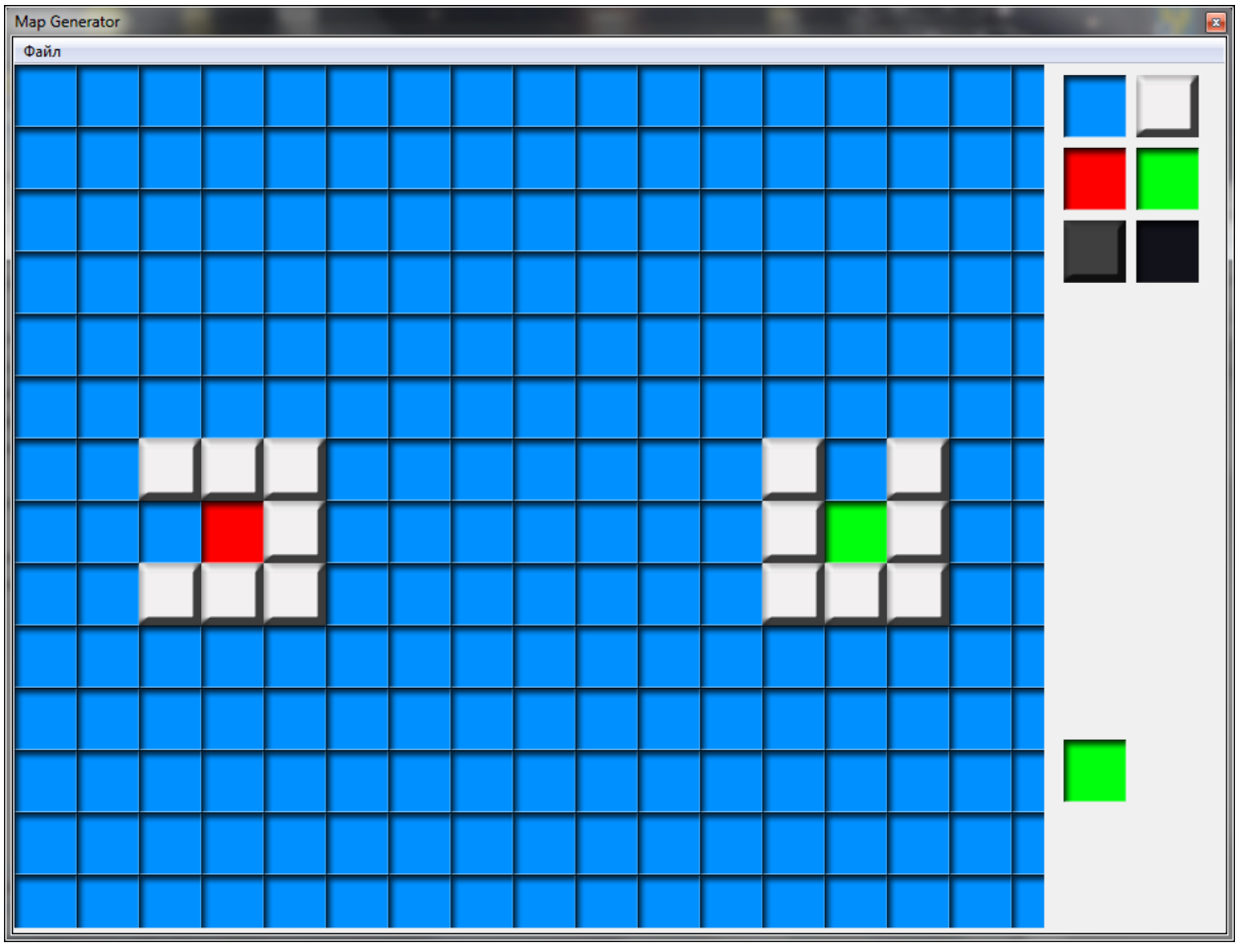


Рисунок 11 – Окно редактора карт

Библиографический список

1. Ростовцев В.С. Оформление курсовых и дипломных проектов для студентов специальности 230101 [Текст] / В.С. Ростовцев, С.Д. Блинова. – Киров: Изд-во ВятГТУ, 2006. – 39 с.
2. Девять сервисов для обучения программированию [Электронный ресурс] / VC. RU, 2014- . − Режим доступа : https://vc.ru/p/code, свободный. − Загл. с экрана.