МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

КАФЕДРА ИНФОРМАТИКИ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по курсу «Операционные системы и сети»

Тема: «Перехват и протоколирование сетевого трафика»

Выполнил  
студент гр. 453504  
Ровдо Н.Р.

Проверил  
Пашук А.В.

Минск 2016

**Содержание**

1. Введение

2. Описание игры(художественное)

3. Выбор языка

4. Концепция

5. Методика работы с программным средством

5. Описание структуры классов

6. Детали реализации

7. Заключение

8. Список использованной литературы

**1. Введение.**

Аркада (англ. arcade) — распространённый в постсоветской индустрии компьютерных игр термин, обозначающий компьютерные игры с нарочно примитивным игровым процессом. В мировой практике аркадами называются игры для аркадных игровых автоматов. Это не отдельный жанр игр, а скорее игровое направление. Компьютерная игра называется «аркадной» в том случае, если она напрямую портирована с автомата или же схожа по концепции с играми для автоматов. Например, к аркадным играм относятся все проекты жанров «файтинг» (fighting), часть игр жанра «гонки» (racing), часть игр жанра «шутер» (shooter). К ним никогда не относятся ролевые игры, симуляторы (кроме т. н. «танцевальных симуляторов»), стратегии.

**2.Описание игры(художественное)**

Далёкое будущее. Ресурсы Земли выработаны до предела. И от корпорации в космос отправляется экспедиция “Deep Decent”, цель которой - вернуться с трюмами, набитыми минералами. Вы являетесь водителем профессиональной буровой машины. Вы должны добывать руду из глубоких недр чужой планеты. Добывайте минералы, сдавайте их, улучшайте буровую машину и покупайте бонусы. Но не забывайте следить за уровнем топлива. Если оно иссякнет, защитные системы отключатся, и машина развалится на куски!

**3.Выбор языка**

Python — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций.

Python поддерживает несколько парадигм программирования, в том числе структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное. Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных. Код в Python организовывается в функции и классы, которые могут объединяться в модули (они в свою очередь могут быть объединены в пакеты).

По сравнению с компилирующими или строго типизированными языками, такими как C, C++ и Java, Python во много раз повышает производительность труда разработчика. Объем программного кода на языке Python обычно составляет треть или даже пятую часть эквивалентного программного кода на языке C++ или Java. Это означает меньший объем ввода с клавиатуры, меньшее количество времени на отладку и меньший объем трудозатрат на сопровождение. Кроме того, программы на языке Python запускаются сразу же, минуя длительные этапы компиляции и связывания, необходимые в некоторых других языках программирования, что еще больше увеличивает производительность труда программиста.

Будучи удачно спроектированным языком программирования Python прекрасно подходит для решения реальных задач из разряда тех, которые разработчикам приходится решать ежедневно. Он используется в самом широком спектре применений – и как инструмент управления другими программными компонентами, и для реализации самостоятельных программ.

Платформа Pygame — набор модулей (библиотек) языка программирования Python, предназначенный для написания компьютерных игр и мультимедиа-приложений. Pygame базируется на мультимедийной библиотеке SDL.

**4. Концепция и правила игры.**

Вы играете за водителя мобильной буровой установки. Основная задача игры добыть максимум ресурсов на максимальной глубине. В ходе прохождения игроку предстоит столкнуться с определёнными сложностями, а именно: c перегревом, малым количеством топлива и чувствительной обшивкой.

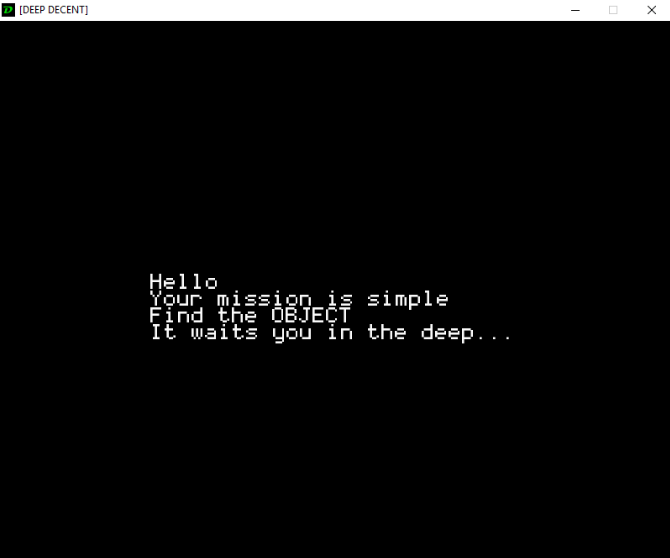
Игрока приветствует заставка, в которой сообщается о особом объекте на глубине, который необходимо доставить на поверхность.

Рисунок Начальная заставка

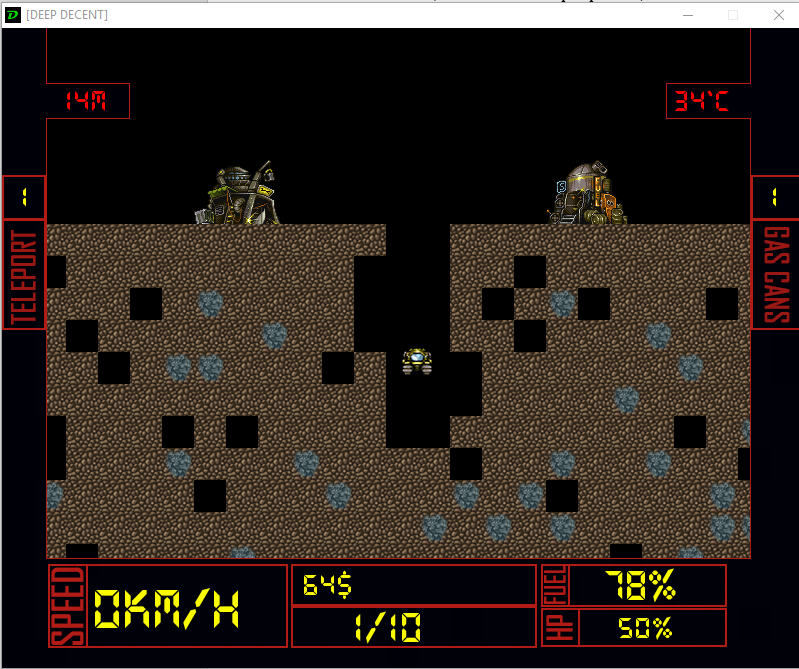
Далее игроку предоставляется управление

Рисунок Начало игры

Интерфейс управления:

* ↑↓→← - Перемещение
* G – использование баллона с топливом
* T – использование телепорта
* Интерфейс установки:
* Сверху слева – текущая глубина
* По центу слева – количество телепортов
* Сверху справа – температура фюзеляжа
* По центру справа - количество баллонов с топливом

Нижняя панель cлева – направо-сверху-вниз:

* Скорость передвижения
* Заработанные деньги
* Склад (занятая часть/всего)
* Количество топлива в процентах
* Количество здоровья в процентах



Рисунок Нижняя часть интерфейса

Игрок спускается ниже, на него начинает действовать определённые условия. Чем ниже спуск, тем жарче, тем более дорогие ресурсы можно добыть, тем больше топлива необходимо, чтобы вернуться назад невредимым.

Игрок должен иметь в виду, что если температура фюзеляжа превысит критическую ( <75 °C), то двигатель установки начнёт работать во вред и количество очков прочности начнёт снижаться. Кроме того установка оснащена двигателем для подъёма и посадки. Игроку необходимо помнить о мягкой посадке, т.к. при падении очки прочности также снижаются. Склад бурильной установки также ограничен, по его истощению игрок не сможет продолжить бурение и ему придётся возвращаться наверх на станцию для сдачи груза и дозаправки.

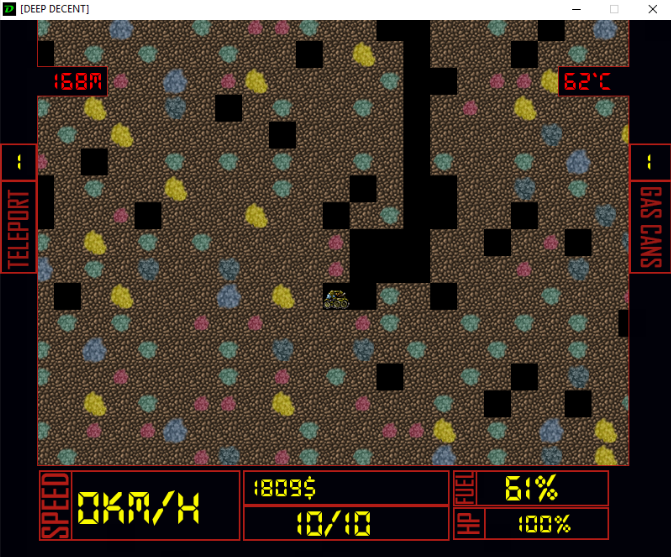


Рисунок Слева-направо: база улучшения и ремонтно-заправочная станция

Для ремонта и дозаправки на поверхности присутствует специальная станция. Игроку достаточно добраться до неё, как топливо и фюзеляж постепенно будут восстановлены до ста процентов. (на рисунке слева)

Для улучшений на поверхности находится база улучшения. На ней игрок может сделать апгрейд системы охлаждения, увеличить размер грузового отсека, объём перевозимого топлива, а также повысить прочность фюзеляжа. Кроме всего прочего в магазине можно приобрести одноразовые телепорты и доп. баки с топливом. Цены приведены на рисунке.

Рисунок Меню базы улучшения



В игре присутствуют различные минералы для добычи. Чем глубже спуск, тем крупнее добыча

Рисунок В процессе игры

|  |  |  |  |
| --- | --- | --- | --- |
|  | Название минерала | Появляется на глубине | Цена за блок |
| D:\LABS\BSUIR_labs\4cem\OOP\py\basic_ground_tileset.png | Обычная земля | 1 | 1$ |
| D:\LABS\BSUIR_labs\4cem\OOP\py\basic_ferrum_tileset.png | Железо | 1 | 50$ |
| D:\LABS\BSUIR_labs\4cem\OOP\py\basic_silver_tileset.png | Никель | 50 | 150$ |
| D:\LABS\BSUIR_labs\4cem\OOP\py\basic_gold_tileset.png | Золото | 150 | 500$ |
| D:\LABS\BSUIR_labs\4cem\OOP\py\basic_ruby_tileset.png | Рубин | 400 | 2500$ |
| D:\LABS\BSUIR_labs\4cem\OOP\py\basic_malahit_tileset.png | Малахит | 250 | 800$ |
| D:\LABS\BSUIR_labs\4cem\OOP\py\basic_alien_tileset.png | Объект | ??? | inf |

На определённой глубине находится Объект – особая технология, главная цель игры. При добыче буровая установка будет инфицирована, после чего управление будет инвертировано, что усложнит возврат наверх. Тогда и будет закончена игра.

**6.Детали реализации**

**Иерархия классов**

# *Class Hierarchy*

* [pause\_menu.Menu](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\pause_menu.Menu-class.html)
* [upgrade\_menu.Menu](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\upgrade_menu.Menu-class.html)
* [menu.Menu](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\menu.Menu-class.html)
* **object**: The most base type
  + [main.Camera](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\main.Camera-class.html)
  + [pygame.font.Font](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\pygame.font.Font-class.html):
  + [pyganim.PygAnimation](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\pyganim.PygAnimation-class.html)
  + [pyganim.PygConductor](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\pyganim.PygConductor-class.html)
  + [pygame.Rect](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\pygame.Rect-class.html)
  + **pygame.sprite.Sprite**:
    - [block.AlienBlock](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\block.AlienBlock-class.html)
    - [block.Border](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\block.Border-class.html)
    - [block.FerrumBlock](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\block.FerrumBlock-class.html)
    - [block.FuelTriggerBlock](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\block.FuelTriggerBlock-class.html)
    - [block.GasolineStationBlock](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\block.GasolineStationBlock-class.html)
    - [block.GoldBlock](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\block.GoldBlock-class.html)
    - [block.MalahitBlock](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\block.MalahitBlock-class.html)
    - [miner.Miner](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\miner.Miner-class.html)
    - [block.NullBlock](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\block.NullBlock-class.html)
    - [block.Platform](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\block.Platform-class.html)
    - [block.RubyBlock](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\block.RubyBlock-class.html)
    - [block.SilverBlock](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\block.SilverBlock-class.html)
    - [block.UpdateTriggerBlock](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\block.UpdateTriggerBlock-class.html)
    - [block.UpgradeStationBlock](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\block.UpgradeStationBlock-class.html)
    - [block.VirusBlock](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\block.VirusBlock-class.html)
    - [block.WallpaperBlock](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\block.WallpaperBlock-class.html)
  + [pygame.Surface](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\pygame.Surface-class.html):
* [pause\_menu.Menu.rectang](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\pause_menu.Menu.rectang-class.html)
* [upgrade\_menu.Menu.rectang](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\upgrade_menu.Menu.rectang-class.html)
* [menu.Menu.rectang](file:///C:\Users\%D0%94%D0%B0%D0%BD%D0%B8%D0%BB%D0%B0%20%D0%91%D0%BE%D0%B4%D1%80%D0%BE%D0%B2\Desktop\menu.Menu.rectang-class.html)

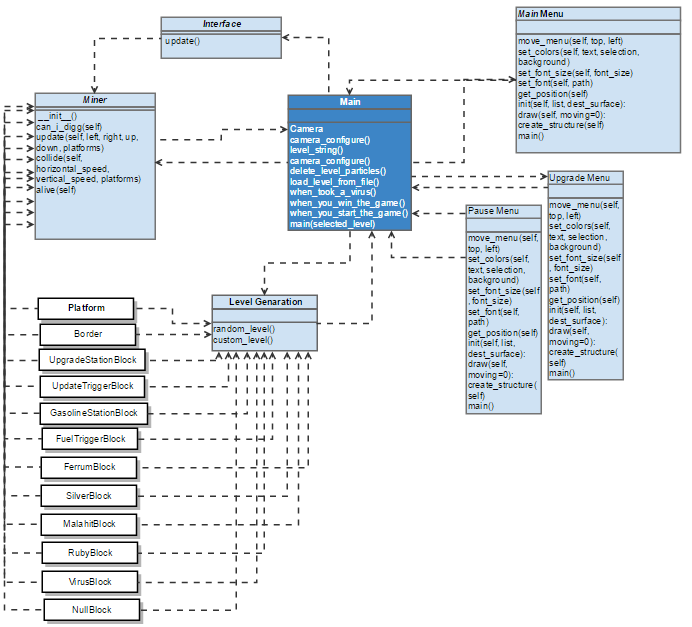


Рисунок Диаграмма классов

**Краткое описание основных классов игры**

* Menu – класс, который запускается при старте игры
* Miner – класс, отвечающий за обработку всех действий, связанных с главным героем: столкновения, управление, взаимодействие.
* Camera – класс, обрабатывающий положение камеры около игрока.
* Block\* - классы – блоки с которыми сталкивается игрок

**Реализация методов:**

**Метод Collide (отвечает за столкновения с объектами)**

def collide(self, horizontal\_speed, vertical\_speed, platforms):

for p in platforms:

if sprite.collide\_rect(self, p):

if vertical\_speed > 0:

if not isinstance(p, block.NullBlock):

if isinstance(p, block.FuelTriggerBlock):

if self.isinfected:

main.when\_you\_win\_the\_game()

self.isinfected = False

# percentage\_lost\_fuel = self.current\_fuel/self.max\_fuel

# percentage\_lost\_health = self.health/100

if self.current\_fuel/self.max\_fuel <= 0.99 or self.health/self.max\_fuel <= 0.99:

# self.current\_fuel = self.max\_fuel

if self.current\_fuel < self.max\_fuel:

self.current\_fuel += 30

if self.current\_fuel>self.max\_fuel:

self.current\_fuel = self.max\_fuel

if self.current\_capacity > 0:

self.current\_capacity -= 1

# self.current\_capacity = 0

if self.health < self.max\_health:

self.health += 1

# self.cash -= round(percentage\_lost\_fuel\*200)

if isinstance(p, block.UpdateTriggerBlock):

data = upgrade\_menu.main(self.cash)

self.cash = data[0]

if data[1] == "f":

self.max\_fuel \*= 2

if data[1] == "a":

self.max\_health \*= 2

if data[1] == "g":

self.max\_capacity \*= 2

if data[1] == "c":

self.cooling\_index += 1

if data[1] == "g":

self.gasoline\_tank += 1

if data[1] == "t":

self.teleports += 1

self.alive()

# self.cash -= round(percentage\_lost\_health\*1000)

# print p.

if self.digging\_vertical and \

not isinstance(p, block.Border) and \

not isinstance(p, block.FuelTriggerBlock) and \

not isinstance(p, block.GasolineStationBlock) and \

not isinstance(p, block.UpgradeStationBlock) and\

not isinstance(p, block.UpdateTriggerBlock):

self.digging\_vertical = False

self.cash += p.value

if not isinstance(p, block.Platform):

self.current\_capacity += 1

if isinstance(p, block.VirusBlock):

self.isinfected = True

print "I infected"

self.current\_capacity = self.max\_capacity

main.when\_took\_a\_virus()

main.delete\_level\_particles(p)

else:

self.rect.bottom = p.rect.top

self.onGround = True

self.vertical\_speed = 0

if vertical\_speed < 0:

if not isinstance(p, block.NullBlock):

self.rect.top = p.rect.bottom

self.vertical\_speed = 0

if horizontal\_speed > 0:

if not isinstance(p, block.NullBlock):

self.rect.right = p.rect.left

if self.digging\_horizontal and\

not isinstance(p, block.Border) and\

not isinstance(p, block.FuelTriggerBlock) and\

not isinstance(p, block.GasolineStationBlock) and \

not isinstance(p, block.UpgradeStationBlock) and\

not isinstance(p, block.UpdateTriggerBlock):

self.digging\_horizontal = False

self.cash += p.value

if not isinstance(p, block.Platform):

self.current\_capacity += 1

if isinstance(p, block.VirusBlock):

self.isinfected = True

print "I infected"

self.current\_capacity = self.max\_capacity

main.when\_took\_a\_virus()

main.delete\_level\_particles(p)

if horizontal\_speed < 0:

if not isinstance(p, block.NullBlock):

self.rect.left = p.rect.right

if self.digging\_horizontal and \

not isinstance(p, block.Border) and \

not isinstance(p, block.FuelTriggerBlock) and \

not isinstance(p, block.GasolineStationBlock) and \

not isinstance(p, block.UpgradeStationBlock) and\

not isinstance(p, block.UpdateTriggerBlock):

self.digging\_horizontal = False

self.cash += p.value

if not isinstance(p, block.Platform):

self.current\_capacity += 1

if isinstance(p, block.VirusBlock):

self.isinfected = True

print "I infected"

self.current\_capacity = self.max\_capacity

main.when\_took\_a\_virus()

main.delete\_level\_particles(p)

**Метод Main(основной while 1 метод)**

def main(selected\_level):

pygame.init() # Инициация PyGame, обязательная строчка

screen = pygame.display.set\_mode(display\_dict) # Создаем окошко

timer = pygame.time.Clock()

bg.fill(Color(background\_color))

hero = miner.Miner(350, 265)

left = right = down = up = False

entities.add(hero)

# b - border

# g - gasoline

# - - ground

font = pygame.font.Font('data/coders\_crux/coders\_crux.ttf', 72)

# выводим надпись

text = font.render("loading...", 1, (250, 250, 250))

screen.blit(text, (300, 300))

pygame.display.update()

if selected\_level == "generate":

level = level\_generation.random\_level()

level\_string(level)

total\_level\_width = len(level[0]) \* platform\_width

total\_level\_height = len(level) \* platform\_height # высоту

if selected\_level == "test":

level = level\_generation.custom\_level()

level\_string(level)

total\_level\_width = len(level[0]) \* platform\_width

total\_level\_height = len(level) \* platform\_height # высоту

if selected\_level == "load":

hero = load\_level\_from\_file()

total\_level\_height = 150 \* platform\_height

total\_level\_width = 45 \* platform\_width

screen.blit(bg, (0, 0))

when\_you\_start\_the\_game()

previous\_speed = 0

camera = Camera(camera\_configure, total\_level\_width, total\_level\_height)

pygame.mixer.music.load('ena.mp3')

pygame.mixer.music.play(0)

while 1:

timer.tick(100)

for e in pygame.event.get():

if e.type == QUIT:

raise (SystemExit, "QUIT")

if e.type == KEYDOWN and e.key == K\_UP:

up = True

hero.current\_fuel -= 3

if e.type == KEYUP and e.key == K\_UP:

up = False

if e.type == KEYDOWN and e.key == K\_LEFT:

left = True

hero.current\_fuel -= 2

if e.type == KEYDOWN and e.key == K\_RIGHT:

right = True

hero.current\_fuel -= 2

if e.type == KEYUP and e.key == K\_RIGHT:

right = False

if e.type == KEYUP and e.key == K\_LEFT:

left = False

if e.type == KEYDOWN and e.key == K\_DOWN:

down = True

hero.current\_fuel -= 5

if e.type == KEYUP and e.key == K\_DOWN:

down = False

if e.type == KEYDOWN and e.key == K\_t:

# menu.init()

if hero.teleports > 0:

hero.alive()

hero.teleports -= 1

if e.type == KEYDOWN and e.key == K\_g:

# menu.init()

if hero.gasoline\_tank > 0:

hero.current\_fuel += 750

hero.gasoline\_tank -= 1

if e.type == KEYDOWN and e.key == K\_ESCAPE:

pause\_menu.main(hero, platforms)

camera.update(hero) # центризируем камеру относительно персонажа

screen.blit(bg, (0, 0))

if abs(previous\_speed-hero.vertical\_speed) >= 15:

hero.health -= 50

previous\_speed = hero.vertical\_speed

for e in entities:

screen.blit(e.image, camera.apply(e))

hero.current\_temperature = (hero.rect.y / 32)\*2 /hero.cooling\_index

if hero.current\_temperature > hero.critical\_temperature:

hero.health -= 0.4

if hero.current\_fuel < 0 or hero.health < 1:

font = pygame.font.Font('data/coders\_crux/coders\_crux.ttf', 72)

# выводим надпись

text = font.render("You lose", 1, (250, 250, 250))

lose\_image = image.load("rect.png")

screen.blit(lose\_image, (270, 270))

screen.blit(text, (300, 300))

pygame.display.update()

time.wait(1000)

hero.alive()

hero.current\_fuel = 1000

hero.health = 100

deep = ((hero.rect.y / 32)-10)\*2

if hero.isinfected:

hero.update(right, left, down, up, platforms)

else:

hero.update(left, right, up, down, platforms)

interface.update(screen,

deep,

hero.vertical\_speed,

hero.current\_capacity,

hero.max\_capacity,

hero.health,

hero.max\_health,

hero.current\_temperature,

hero.current\_fuel,

hero.max\_fuel,

hero.cash,

hero.teleports,

hero.gasoline\_tank)

pygame.display.update()

**Метод random\_level(случайная генерация уровня)**

def random\_level():

random\_level = []

level\_width = 30

level\_height = 150

line = ""

for \_ in range(level\_width):

line += "b"

random\_level.append(line)

for i in range(level\_height):

line = ""

line += "b"

for j in range(level\_width):

if i <= 10:

line += " "

if i == 11:

if j == 15:

line += "G"

if j == 5:

line += "U"

if j == 25:

line += "v"

else:

line += " "

if i == 12:

line += " "

if i == 13:

if j == 15 or j == 16 or j == 17:

line += "g"

elif j == 5 or j == 6 or j == 7:

line += "u"

else:

line += "-"

if 13 < i < 35:

ground\_or\_nothing = randint(0, 100)

if ground\_or\_nothing < 10:

line += " "

if 10< ground\_or\_nothing < 20:

line += "f"

else:

line += "-"

if 35 <= i < 65:

ground\_or\_nothing = randint(0, 100)

if ground\_or\_nothing < 10:

line += " "

if 10 <= ground\_or\_nothing < 20:

line += "s"

if 20 <= ground\_or\_nothing < 30:

line += "f"

else:

line += "-"

if 65 <= i < 75:

ground\_or\_nothing = randint(0, 100)

if ground\_or\_nothing < 10:

line += " "

if 10 <= ground\_or\_nothing < 20:

line += "s"

if 20 <= ground\_or\_nothing < 30:

line += "f"

if 30 <= ground\_or\_nothing < 50:

line += "o"

else:

line += "-"

if 75 <= i < 80:

ground\_or\_nothing = randint(0, 100)

if ground\_or\_nothing < 10:

line += " "

if 10 < ground\_or\_nothing < 15:

line += "s"

if 15 <= ground\_or\_nothing < 20:

line += "f"

if 20 <= ground\_or\_nothing < 50:

line += "o"

if 50 <= ground\_or\_nothing < 60:

line += "m"

else:

line += "-"

if i > 80:

ground\_or\_nothing = randint(0, 100)

if ground\_or\_nothing < 10:

line += " "

if 10 <= ground\_or\_nothing < 15:

line += "s"

if 15 <= ground\_or\_nothing < 20:

line += "f"

if 20 <= ground\_or\_nothing < 30:

line += "o"

if 50 <= ground\_or\_nothing < 70:

line += "m"

if 70 <= ground\_or\_nothing < 80:

line += "r"

else:

line += "-"

line += "b"

random\_level.append(line)

line = ""

for \_ in range(level\_width):

line += "b"

random\_level.append(line)

return random\_level

**Meтод level\_string(обработка уровня)**

def level\_string(level):

x = y = 0

for row in level:

for col in row:

if col == "-":

pf = block.Platform(x, y)

entities.add(pf)

platforms.append(pf)

if col == "b":

pf = block.Border(x, y)

entities.add(pf)

platforms.append(pf)

if col == "g":

pf = block.FuelTriggerBlock(x, y)

entities.add(pf)

platforms.append(pf)

if col == "G":

pf = block.GasolineStationBlock(x, y)

entities.add(pf)

platforms.append(pf)

if col == "u":

pf = block.UpdateTriggerBlock(x, y)

entities.add(pf)

platforms.append(pf)

if col == "U":

pf = block.UpgradeStationBlock(x, y)

entities.add(pf)

platforms.append(pf)

if col == "f":

pf = block.FerrumBlock(x, y)

entities.add(pf)

platforms.append(pf)

if col == "s":

pf = block.SilverBlock(x, y)

entities.add(pf)

platforms.append(pf)

if col == "m":

pf = block.MalahitBlock(x, y)

entities.add(pf)

platforms.append(pf)

if col == "r":

pf = block.RubyBlock(x, y)

entities.add(pf)

platforms.append(pf)

if col == "o":

pf = block.GoldBlock(x, y)

entities.add(pf)

platforms.append(pf)

if col == "v":

pf = block.VirusBlock(x, y)

entities.add(pf)

platforms.append(pf)

else:

pf = block.NullBlock(x, y)

entities.add(pf)

platforms.append(pf)

x += platform\_width

y += platform\_height

x = 0

**Метод delete\_level\_particles(удаление частиц)**

def delete\_level\_particles(digged\_block):

platforms.remove(digged\_block)

entities.remove(digged\_block)

x = digged\_block.x

y = digged\_block.y

digged\_block = block.NullBlock(x, y)

platforms.append(digged\_block)

entities.add(digged\_block)

**7. Заключение.**

По итогу: была написана 2D игра, использующая технологии языка Python, а в частности библиотеку pygame. В ходе написания курсового проекта были рассмотрены такие инструменты для создания игрового контента как анимации, спрайты, паттерн “Strategy” и др. В качестве сильных сторон проекта хочу выделить расширяемость: для продолжения истории, расширения карты, добавления доп. возможностей программисту не потребуется глобальный рефакторинг исходного кода, а только доп. логика. Рассматривая минусы можно отметить, зависимость от аппаратного обеспечения: игра требует немалых ресурсов для обработки большого количества спрайтов на большой локации, зависимость от программного обеспечения: для запуска игры на разных платформах необходимо наличие предустановленного пакета языка Python и модуля pygame, что усложняет запуск для рядового пользователя.

Весь код доступен по [ссылке](https://github.com/keipa/DeepDecent) (<https://github.com/keipa/DeepDecent>), позже на официальном сайте библиотеки pygame (pygame.org).

**8. Список использованной литературы**

1. Лутц М. - Изучаем Python (4-е издание)- 2011

2. Лутц М. - Программирование на Python, 4-е издание в 2-х томах

3. pygame.org

4. Wikipedia.org

5. docs.python.org