МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ федеральное государственное автономное образовательное учреждение высшего образования «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ» КАФЕДРА № 43

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

КУРСОВОЙ ПРОЕКТ

ЗАЩИЩЕН С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Доц., к.т.н |  |  |  | А.В. Туманова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ  Разработка программы |
| «TANK-MASTERS: CLASSIC» |
| по курсу: ОСНОВЫ ПРОГРАММИРОВАНИЯ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4134к |  |  |  | Столяров Н.С. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

**Описание**

TANK-MASTERS: CLASSIC – это шутер с видом сверху (top-down). Написанный на языке программирования Python 3 с использованием PyOpenGL (для рендеринга).

**Интерфейс**

Был выбран большой размер кнопок для интерфейса т.к. предполагалось что данный проект будет портирован на мобильные устройства, чего не случилось из-за специфичной работы openGL под ними.



**Различные страницы меню**

* Выбор карт и режимов игры (на данный момент доступен только Death Match)

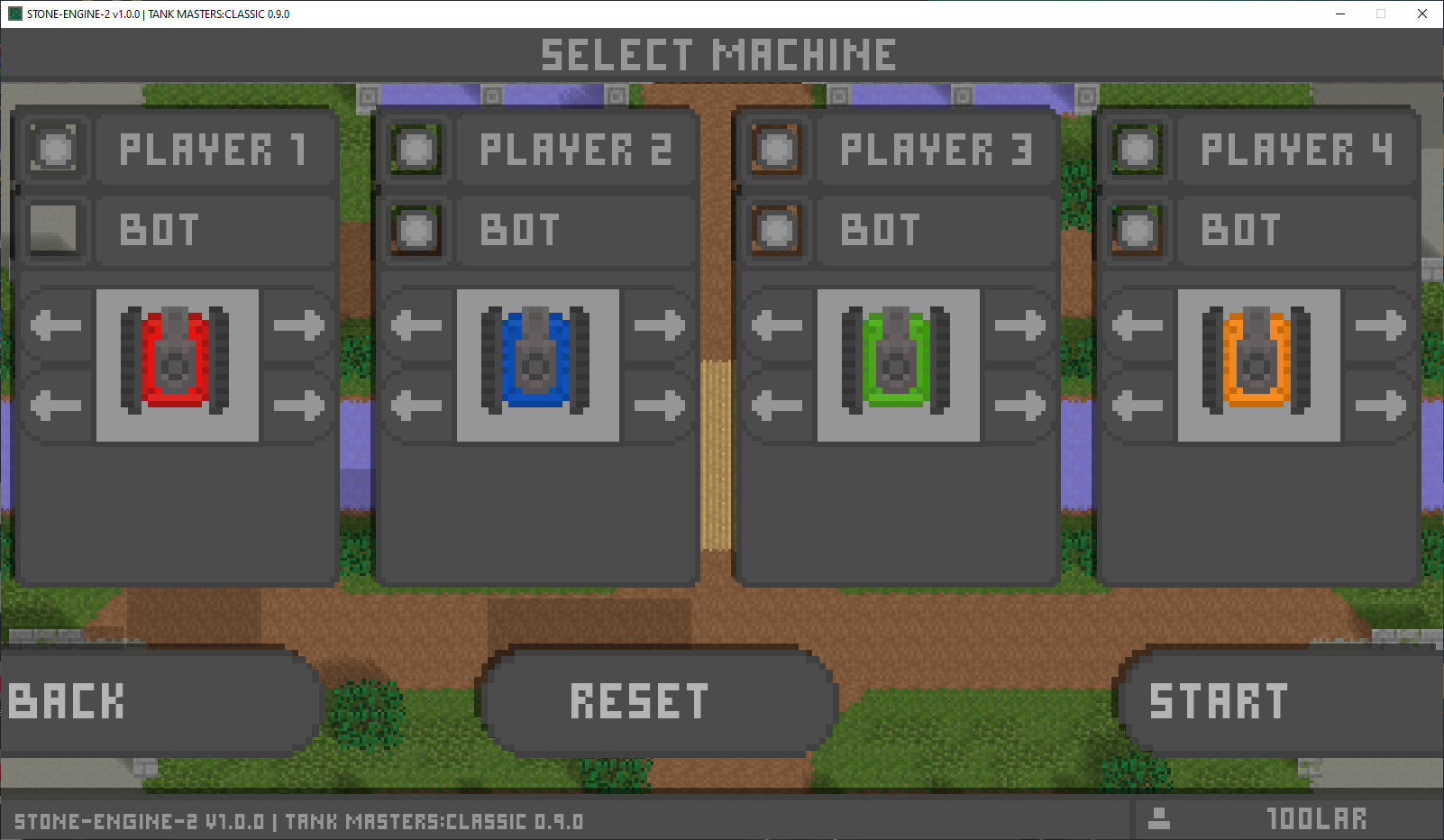


Рисунок 2 - рпропор

* Настройка игровых правил и погоды



* Выбор параметров танчиков.



**Игровой процесс**

Суть стандартного режима игры (Death Match) – это уничтожить как можно больше противников за определённое время. По окончанию таймера игра останавливается и подводятся итоги.

**Редактор карт**

Простой редактор карт позволяет послойно “рисовать” карту и расставлять точки.

Файлы карт сохраняются по пути ~/maps/arcade



**Рендеринг карты**

Карта строится из тайлов (набора небольших картинок, в данном случае с разрешением 8х8)

Каждый тайл (квадрат 8х8 пикселей) рисовать не экономично для ресурсов ПК. Следовательно было принято решение которое объединяет все тайлы в одну большую картинку. И таким образом получилось сэкономить ресурсы. Ниже приведён пример такого решения.

# создаём пустые изображения для слоёв карты

self.temp\_image\_floor = Image.new('RGBA', (self.world\_size[0] \* self.size, self.world\_size[1] \* self.size))

def set\_floor(self):

print("SET FLOOR")

for y in range(self.world\_size[1]):

for x in range(self.world\_size[0]):

block = self.map\_floor[self.get\_block\_num(x, y)].split('.')

if block[0] != 'none':

if get\_obj\_display('game\_settings').snow:

try:

self.temp\_image\_floor.paste(self.floor\_snow\_blocks\_img[block[0]].rotate(int(block[1])), (x \* self.size, y \* self.size))

except:

self.temp\_image\_floor.paste(self.floor\_blocks\_img[block[0]].rotate(int(block[1])), (x \* self.size, y \* self.size))

else:

self.temp\_image\_floor.paste(self.floor\_blocks\_img[block[0]].rotate(int(block[1])), (x \* self.size, y \* self.size))

def import\_images(self):

print('IMPORT IMAGES')

self.image\_floor = PIL\_to\_pyglet(self.temp\_image\_floor, self.scale)

**Баллистика пуль**

В зависимости угла куда дует ветер (направление ветра показана листьями и облаками) меняется направление движение снаряда.

if get\_obj\_display('game\_settings').wind\_bool and bullet[5] != 1:

wind\_deg = get\_obj\_display('game\_settings').wind\_deg

wind\_power = speed\_tick / get\_obj\_display('game\_settings').wind\_power

wind = bullet[3] + wind\_deg

if wind > 180:

wind = wind - 360

if wind < -180:

wind = wind + 360

if wind < 0:

bullet[3] -= wind\_power

elif wind > 0:

bullet[3] += wind\_power

**Ящики снабжения**

При подборе такого ящика может выдаться 1 из 2х положительных эффектов

|  |  |
| --- | --- |
| C:\Users\nikit\AppData\Local\Microsoft\Windows\INetCache\Content.Word\box.png | * Восстановление уровня здоровья до изначального * Щит (держит только 1 попадание из любого калибра) |

Ниже расположен код спавна и коллизии ящиков.

def add\_crate(self):

while True:

rand\_pos = [

random.randint(0, get\_obj\_display('world').world\_size[0]),

random.randint(0, get\_obj\_display('world').world\_size[1])

]

'''check\_pos\_list = [

[0, 0],

[1, 0],

[0, 1],

[1, 1]

]'''

check\_pos\_list = [

[-1, 1],

[0, 1],

[1, 1],

[-1, 0],

[0, 0],

[1, 0],

[-1, -1],

[0, -1],

[1, -1]

]

ok = True

for i in range(len(check\_pos\_list)):

try:

if (

get\_obj\_display('world').get\_wall\_poligon(rand\_pos[0] + check\_pos\_list[i][0], rand\_pos[1] + check\_pos\_list[i][1]) != 'none'

or get\_obj\_display('world').get\_water\_poligon(rand\_pos[0] + check\_pos\_list[i][0], rand\_pos[1] + check\_pos\_list[i][1]) != 'none'

):

ok = False

except:

ok = False

if ok:

rand\_pos = [

get\_obj\_display('world').image\_floor.x + (rand\_pos[0] \* get\_obj\_display('world').size \* get\_obj\_display('world').scale) - get\_obj\_display('world').map\_offs[0],

get\_obj\_display('world').image\_floor.y + ((get\_obj\_display('world').world\_size[1] - rand\_pos[1]) \* get\_obj\_display('world').size \* get\_obj\_display('world').scale) - get\_obj\_display('world').map\_offs[1]

]

self.crates\_list.append([rand\_pos[0], rand\_pos[1]])

break

def update(self):

if get\_obj\_display('game\_settings').crates\_bool:

for i in range(len(self.crates\_list)-1, -1, -1):

self.crate\_poly.pos.x = self.crates\_list[i][0]

self.crate\_poly.pos.y = self.crates\_list[i][1]

for p in get\_obj\_display('players').tanks:

if collision.collide(p.poligon\_body, self.crate\_poly):

if p.health < p.default\_health:

p.health = p.default\_health

else:

p.armor\_bool = True

p.protection = False

self.crates\_list.pop(i)

self.sound.play('upgrade.wav')

self.add\_crate()

break

**Погода**

В настройках игровых правил есть возможность включить дождь и снег. Дождь ухудшает видимость, а снег в добавок покрывает всю карту снегом.

<<Дождь>>

<<Снег>>



Код релизации дождя

**class** **weather():**

**def** get\_speed\_by\_deg\_sp**(**self**,** deg**,** speed**):**

#deg += 90

**if** deg **<** 0**:**

deg **=** 180 **+** **(**180 **+** deg**)**

**if** deg **>=** 360**:**

deg **=** 0

a **=** speed**/**math**.**sin**(**math**.**radians**(**90**))**

x **=** a **\*** math**.**sin**(**math**.**radians**(**deg**))**

y **=** a **\*** math**.**sin**(**math**.**radians**(**180 **-** 90 **-** deg**))**

**return** x**,** y

**def** \_\_init\_\_**(**self**):**

**try:**

self**.**deg **=** 45

self**.**image\_rain **=** Image**.**new**(**'RGBA'**,** **(**settings**.**width**\***2**,** settings**.**height**\***2**))**

image **=** Image**.open(**'assets/img/world/rain/' **+**

**(**'rain' **if** **not** get\_obj\_display**(**'game\_settings'**).**snow **else** 'snow'**)**

**+** '.png'**).**resize**(((**settings**.**height**//**50**)//**8**,** **(**settings**.**height**//**50**)),** Image**.**NEAREST**).**convert**(**"RGBA"**)**

self**.**speed **=** settings**.**height**/**80 **if** **not** get\_obj\_display**(**'game\_settings'**).**snow **else** settings**.**height**/**200

self**.**speed\_x**,** self**.**speed\_y **=** self**.**get\_speed\_by\_deg\_sp**(**self**.**deg**,** self**.**speed**)**

self**.**delay **=** 1**/(**settings**.**fps**/**1.6**)**

self**.**time **=** time**.**perf\_counter**()** **+** self**.**delay

self**.**max\_rain **=** 1000 **if** **not** get\_obj\_display**(**'game\_settings'**).**snow **else** 10000

**for** \_ **in** **range(**self**.**max\_rain**):**

x **=** random**.**randrange**(**0**,** settings**.**width**\***2**)**

y **=** random**.**randrange**(**0**,** settings**.**height**\***2**)**

self**.**image\_rain**.**paste**(**image**,**

**(**x**,** y**)**

**)**

raw\_image **=** self**.**image\_rain**.**tobytes**()**

self**.**rain\_arr **=** **[]**

**for** i **in** **range(**3**):**

image\_rain **=** pyglet**.**image**.**ImageData**(**self**.**image\_rain**.**width**,** self**.**image\_rain**.**height**,** 'RGBA'**,** raw\_image**,** pitch**=-**self**.**image\_rain**.**width **\*** 4**)**

image\_rain**.**anchor\_x **=** image\_rain**.**width **//** 2

image\_rain**.**anchor\_y **=** image\_rain**.**height **//** 2

self**.**rain\_arr**.**append**([**pyglet**.**sprite**.**Sprite**(**image\_rain**,** settings**.**width**//**4**,** settings**.**height**//**2**),** **(**i **\*** settings**.**width**),(** i **\*** settings**.**height**)])**

self.rain\_arr[i][0].rotation = self.deg

self.rain\_arr[i][0].x = self.rain\_arr[i][1]

self.rain\_arr[i][0].y = self.rain\_arr[i][2]

except:

print("FATAL ERROR: " + str(traceback.format\_exc()))

def update(self):

if get\_obj\_display('game\_settings').rain and not get\_obj\_display('game\_settings').pause:

if self.time <= time.perf\_counter():

for i in range(len(self.rain\_arr)):

self.rain\_arr[i][1] -= self.speed\_x

self.rain\_arr[i][2] -= self.speed\_y

if (self.rain\_arr[i][1]) < -(settings.width) or (self.rain\_arr[i][2]) < -(settings.height):

self.rain\_arr[i][1] = int(settings.width\*1.2)

self.rain\_arr[i][2] = int(settings.width\*1.2)

self.rain\_arr[i][0].x = self.rain\_arr[i][1]

self.rain\_arr[i][0].y = self.rain\_arr[i][2]

self.time = time.perf\_counter() + self.delay

def draw(self):

if get\_obj\_display('game\_settings').rain:

for rain in self.rain\_arr:

drawp(rain[0])

**Исходный код:**

Весь остальной код и ресурсы игры расположены на репозитории (>8000 строк кода не уместить в word документе)

GIT: <https://github.com/4LAR/TANK-MASTERS-CLASSIC>

## **Список литературы**

1. Ключарев А.А., Матьяш В.А., Щекин С.В. Структуры и алгоритмы обработки данных: Учебное пособие / СПбГУАП. СПб., 2004.

2. Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие / Колдаев В.Д; Под ред. проф.Л.Г. Гагариной - М.: ИД ФОРУМ, НИЦ ИНФРА-М, 2016. - 416 с. <http://znanium.com/catalog.php?bookinfo=537513>

6. Кнут, Д. Искусство программирования [Текст] = The art of computer programming : [в 3 т.]. Т. 1. Основные алгоритмы / Д. Кнут ; ред. Ю. В. Козаченко. - 3-е изд. - М. : Вильямс, 2014. - 720 с.

8. Вирт, Н Алгоритмы и структуры данных. Новая версия для Оберона + CD [Текст] / Н. Вирт ; пер. Д. Б. Подшивалов. - 2-е изд., испр. - М. : ДМК Пресс, 2012. - 272 с.