**Пояснительная записка к приложению «2D-шутер с видом сверху»**

**Цель проекта:** создать игру – 2D-шутер, в которой игрок должен проходить уровни, зачищая комнаты в здании и убивая противников

**Исполнитель:** Ятченко Кирилл Вячеславович, Русанов Евгений Васильевич

**Используемые библиотеки:**

pygame==2.1.2

Pillow==9.3.0

**В приложении реализованы следующие возможности:**

* При входе открывается окно регистрации, в котором пользователь вводит свое имя и пароль, по которым он сможет зайти в следующий раз
* После входа открывается основное меню, в котором игрок может выбрать уровень
* После выбора уровня загружается сам уровень, теперь цель игрока – уничтожить всех противников, чтобы пройти уровень
* В случае удачного(или неудачного) завершения игры показывается финальное окно с результатом игрока и набранными очками

**Реализованные функции:**

**Pic\_to\_map() – по .png картинке строит карту и возвращает массив со стенами и другими обьектами**

**Translation\_coordinates() – переводит координаты из абсолютных в относительные**

**Data\_translation() – считывает данные с массива стен**

**Defining\_intersection() – проверяет, пересекается ли данная точка со стенами**

**Draw\_flashlight() – рисует область, освещенную фонариком**

**Spawn\_enemies() – расставляет врагов**

**Search()- поиск нужной информации в БД**

**Addendum()-добавление аккаунта**

**Registration()-логика кнопки регистрации**

**Entrance()-логика кнопки входа в аккаунт**

**Password\_check()–проверка пароля на сложность**

**Reading()-считывает нужные данные из БД**

**Separation()–отрисовывает главное меню**

**Victory()–действия которые происходят при победе игрока на уровне**

**Реализованные классы в приложении:**

**Weapon** – основной класс оружия

Методы:

* is\_reloading() – проверяет, перезаряжается ли оружие
* spawn\_bullet() – создает пулю
* reload\_update() – обновляет счетчики перезарядки
* check\_reload\_start() – проверяет, нужно ли начать перезарядку на этом кадре
* update() – общее обновление
* draw\_interface() – отрисовка интерфейса оружия

От Weapon наследуются классы оружия **Ak\_47, Glock, Shotgun** (соответственно классы автомата, пистолета и дробовика)

У **Ak\_47** и **Glock** новых методов нет

У **Shotgun** есть новые методы:

* spawn\_bullet() – создает не 1, а 9 пуль
* reload\_update() и check\_reload\_start() имеют то же назначение, что и у Weapon, но реализованы несколько другим образом, чтобы сымитировать работу дробовика

**Knife** – класс еще одного вида оружия, ножа

Методы:

* update() – общее обновление, в котором происходит проверка на атаку и т.д.
* draw\_interface() – отрисовка интерфейса ножа

**LootBox** – класс ящика с улучшениями

Методы:

* update() – обновление спрайта
* use() – завершение поднятия улучшения
* reset\_timer() – сброс таймера поднятия
* add\_timer() – увеличение таймера
* draw\_open\_progress() – рисует полоску прогресса открытия

**MedkitLootbox** – наследуется от Lootbox, отличается только перегруженным методом use(), который дает игрока 1 аптечку

**Bullet** – класс пули

Методы:

update() – обновление спрайта, перемещение

**ShotgunBullet** – наследуется от Bullet, отличается только наличием в update() ограничения на время существования, чтобы пуля не улетала слишком далеко

**Entity** – основной класс сущности, от которого наследуются классы игрока и врага

* Методы:  
  Анимация:

reset\_reload\_attack() – сбрасывает таймер перезарядки

anim\_reload\_update() – обновление таймеров перезарядки для анимаций

anim\_attack\_update() – обновление таймеров атаки (для анимаций)

anim\_is\_idle\_update() – обновление таймеров стояния на месте (для анимаций)

anim\_is\_moving\_update() – обновление таймеров движения (для анимаций)

all\_anims\_update() – обновление всех анимаций

get\_current\_state() – получение состояния персонажа (для анимаций)

get\_current\_image\_info() – получение текущей картинки игрока

Другое:

take\_damage() – получить урон

move\_entity() - переместиться

draw\_health\_bar() – рисует полоску очков здоровья

determining\_angle() – определяет угол между двумя лучами от сущности до двух точек

beam() – пускает луч (используется для системы видимости и отрисовки фонарика)

**Player** – класс игрока

Методы:

kill() – вызывается на смерти игрока

end\_game() – вызывается на смерти игрока, показывает финальный экран

get\_current\_weapon() – возвращает текущее оружие игрока

heal() – использование аптечки

draw\_interface() – отрисовка интерфейса врага

get\_nearest\_door() – возвращает ближайшую дверь

get\_nearest\_lootbox() – возвращает ближайший ящик с улучшениями

tracing() – трассировка лучей для фонарика

visible\_objects() – обновляет видимость врагов

update() – общее обновление

**Enemy** – класс врага

Методы:

get\_current\_weapon(), get\_current\_state(), all\_anims\_update(), update() – аналогично классу Entity

detection\_player() – проверяет видимость игрока врагом

See(), Lost(), run() – действия врага в зависимости от того, видит враг игрока, не видит, или враг двигается по прямой

observation() – наблюдение за некоторой областью карты

**Wall** – класс стены

Методы:

update() – обновление спрайта

**Door** – класс двери

Методы:

use() – использование двери

change\_image() – изменение текущей картинки двери

update() – обновление спрайта

get\_current\_image() – возвращает текущую картинку двери

**Camera** – класс камеры для перемещения точки зрения вместе с игроком

Методы:

apply() – применить сдвиг к спрайту

update() – обновить положение камеры

**PlayerAnimation** и **EnemyAnimation** – классы-контейнеры анимаций игрока и врага

Методы:

get\_current\_image() – возвращает текущую картинку для заданного спрайта

**MapTexture** – класс изображения карты, которое отрисовывается позади всех сущностей и служит фоном

Методов нет, т.к. данный класс – просто картинка на заднем фоне

**Button\_circle, Button\_rect, InputBox, text\_output**–отвечают за объекты

интерфейса