

**Курс CITB331**

**Програмиране с Python**

***Arcade Space Shooter***

***With PyGame and sockets***

***Денис Нейчев – F83260***

***София, Юни 2018***

*1. Представяне 2*

*2. Цели 2*

*3. Използвани технологии, дизайнерски практики и инструменти 2*

*4. Реализация 3*

*5. Ресурси 3*

# Представяне

Arcade Space Shooter е семпла, мултиплейър игра за двама. Целта на играта е да победиш своя опонент, като като го стреляш с лазерни лъчи и така отнемаш жизнените му точки. Всеки “удар в целта” ,кара противниковия играч да загуби 10 жизнени точки.

# Цели

Целите на проекта са да предостави на играчите развлекателна игра, с която да се забавляват.

# Използвани технологии, дизайнерски практики и инструменти

Главната технология използвана в проекта е езика за програмиране Python v3.6 и наличните в него механизми за работа със сокети. В допълнение е използвана библиотеката PyGame - Cross-platform, библиотека за разработка на игри написана на C . В реализацията, също така е използван външен скрипт, за приемане на текстов вход от потребителя, базиран на PyGame – ezText.py.

Главната дизайнерска практика използвана при реализацията е *Обектно Ориентирано Програмиране* и всички други асоциирани с него добри практики *(наследяване, енкапсулация и тн.).* Също така, до голяма част са спазени правилата наложени от PEP8, конвенцията за писане на python код.

Използваните инструменти са :

* Atom */Hackable text editor/* - Използван за направата на по – голямата част от проекта
* PyCharm */Python IDE developed by JetBrains/* - Използван на по – късен етап за завършване на проекта
* Python-autopep8 */Plugin for Atom/* - Използван за прилагане на правилата за писане на Python наложени от PEP8

# Реализация

Проектът е разделен на две основни части. Първата е клиент – Логика за показване на елементите и контрол на входа, а втората е сървър – Контрол на играта и логически функции.

**Структура на клиент:**

* Assets – *Всички графични елементи използвани в клиента.*
* Models – *Класове използване за дефиниране на всички клиенти, обекти и техните* *поведения.*
  + Entity.py - *Най -ниското ниво клас. Съдържа общи елементи на Player и Projectile.*
  + Message.py – *Използва се за предаване на съобщения на играчите, в случай на събитие или грешка.*
  + Player.py – *Дефинира елементите на играча и неговите поведения.*
  + Projectile.py – *Дефинира елементите на един снаряд*.
* Utils – *Всички помощни функции нужни на клиента, за да работи правилно.*
  + DrawingUtils.py – *Помощни функции, които рисуват по екрана с предварително зададени параметри.*
  + EzText.py – *EzText библиотеката, която се използва за приемане на текстов вход от играча.*
  + GobalVars.py – *За променливи, нуждаещи се от консистентно състояние.*
  + NetworkingUtils.py – *Методи за инициализиране и управление на сокети*.
* Definitions.py – *Общи константи, използвани от кода.*
* Main.py – *Входна точка на клиента.*

**Структура на сървър:**

* ServModels – *Класове използвани за дефинирани на всички сървър обекти и техните поведения.*
  + Match.py – *Дефинира мач (връзка м/у два клиента), пази техните състояния и сокет връзки.*
* Utils - *Всички помощни функции, нужни на сървъра, за да работи правилно.*
  + gameUtils.py – *Функции, нужни на сървъра, за да управлява мач.*
* Definitions.py – *Общи константи, използвани от сървъра.*
* Server.py - *Входна точка на сървъра.*

**Цялостен работен процес на програмата:**

* 1. Играчът влиза в играта и от него се очаква да въведе своя никнейм.
  2. Когато играчът е въвел своя никнейм, клиентът прави връзка със сървъра, и сървъра го слага в списъка с играчи, които търсят опонент. Ако в списъка вече има друг играч, сървъра ги сдвоява в мач и играта започва. Ако няма друг играч, клиента чака до появяването на такъв.
  3. След започване на мача, клиентът комуникира със сървъра на всеки кадър, като първо праща своята позиция и дали е изстрелял нов снаряд.
  4. Сървъра обработва информацията, изпратена от двата играча, обръща координатите им и ги изпраща на техните противници (всеки вижда противника си от горната страна на екрана, за това координатите се обръщат). Информацията, която се изпраща обратно към клиента, се състои от неговите собствени жизнени точки (сървъра може да ги промени ако е настъпила колизия), и обект координатите на противника му и неговите проектили.
  5. Запазвайки, обърнатата информация за единия от играчите и оригиналната информация за другия, проверява за колизия (сървъра си представя единия от играчите като играч едно или “долен” играч, а другия като играч две или “горен играч”). Ако има колизия, отнема нужните точки от ударения играч.
  6. След проверката за колизия, сървърът проверява, дали мача не е приключен. Условията за това са жизнените точки на единия от играчите да са равни на нула . Ако такива условия са на лице, той оповестява резултатите до играчите и приключва мача.

Горния работен процес се повтаря докато една от страните не победи или докато връзката не прекъсне по други причини (сървърът крашне или някой от играчите се изключи от играта.).



Фигура 1 - Връзка между сървъра и клиентите

# Ресурси

* <https://opengameart.org/content/space-shooter-art> - Графични ресурси
* <https://www.pygame.org/docs/> - Документация на Pygame
* <https://e-edu.nbu.bg/course/view.php?id=18714> – Лекциите от курса “Програмиране с Python”
* <https://docs.python.org/2/howto/sockets.html> -Документацията на Python за сокети
* <https://stackoverflow.com/> - Няма нужда от обяснение ☺