Проект по предметот:
Визуелно Програмирање

FINKI Adventures

GITHUB LINK: ***https://github.com/WoodyMKD/FINKI-Adventures***

Членови на тимот:

• *Методија Новковски (171529)*

• *Стефан Тодоровски (171511)*

Професор: *Проф. Д-р Ѓорѓи Маџаров*

Асистент: *М-р Александар Стојменски*

1. Опис на апликацијата

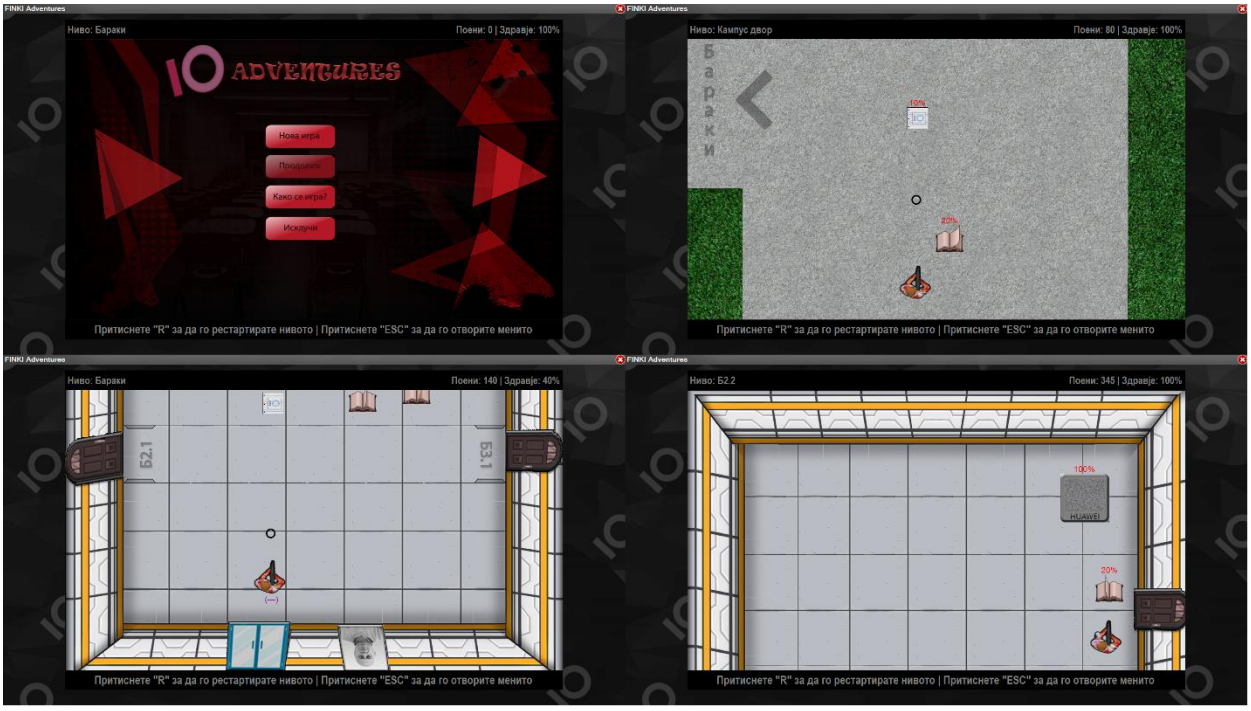
Апликацијата што ја развивме се нарекува “FINKI Adventures” и се работи за класична “Top-down 2D Shooter” игра што целосно ја опишува авантурата која што еден студент ја поминува низ студирањето на ФИНКИ на еден интересен начин.

Главниот играч - протагонист во играта е токму студентот кој што се соочува со различни пречки. Започнувајќи од самиот момент кога ќе пристигне на кампусот, тој е нападнот од многу непријатили во форма на различни книги/скрипти и испитни тетратки кои треба да бидат што побрзо совладани. Ова всушност го симболизира притисокот низ кој минуваат студентите кои постојано се соочени да ги реализираат временски ограничените задолженија зададени од страна на професорите. Временскиот притисок е имплементиран како вертикален скролер на мапата кој го поттикнува играчот побрзо да прогресира низ играта, без притоа да биде „изеден“ од мапата. Надминувајќи ги овие препреки, студентот добива поени и прогресира низ различни нивоа од факултетот за на крај да стигне на завршниот испит каде го чека вистинскиот предизвик.

Додека играчот прогресира низ нивоата на играта, тежината се зголемува така што секое ниво располага со поголем број непријатели од претходното, се додека играчот не пристигне до крајното ниво – испитот, каде се соочува еден непријател кој е потежок за совладување. Целта на играчот е да ги совлада сите непријатели додека не го најде излезот на нивото и да транзитира во наредното ниво.

2. Упатство за користење / играње

На почетокот имаме почетно мени кое ни дозволува да започнеме нова игра или да продолжиме од каде што сме застанале претходно. Откако ќе се кликне едно од овие копчиња се прикажува мапата и играта започнува. Играчот се придвижува со помош на навигациските копчиња или на копчињата: *(W,A,S,D)*, а испукува од пушката со помош на копчето *(Space)*. За да напредува во нивоата, играчот треба да ги елеменира сите непријатели и да го најде патот до следното ниво кое е означено на самата мапа и меѓувремено да пази да не биде "изеден" од мапата која што се движи.



Со притискање на копчето (ESC) се отвора почетното мени, а истотака имплементирана е Debug функција на (TAB) со која се прикажуваат hitbox границите на објектите со црвена боја.

3. Опис на решението на проблемот

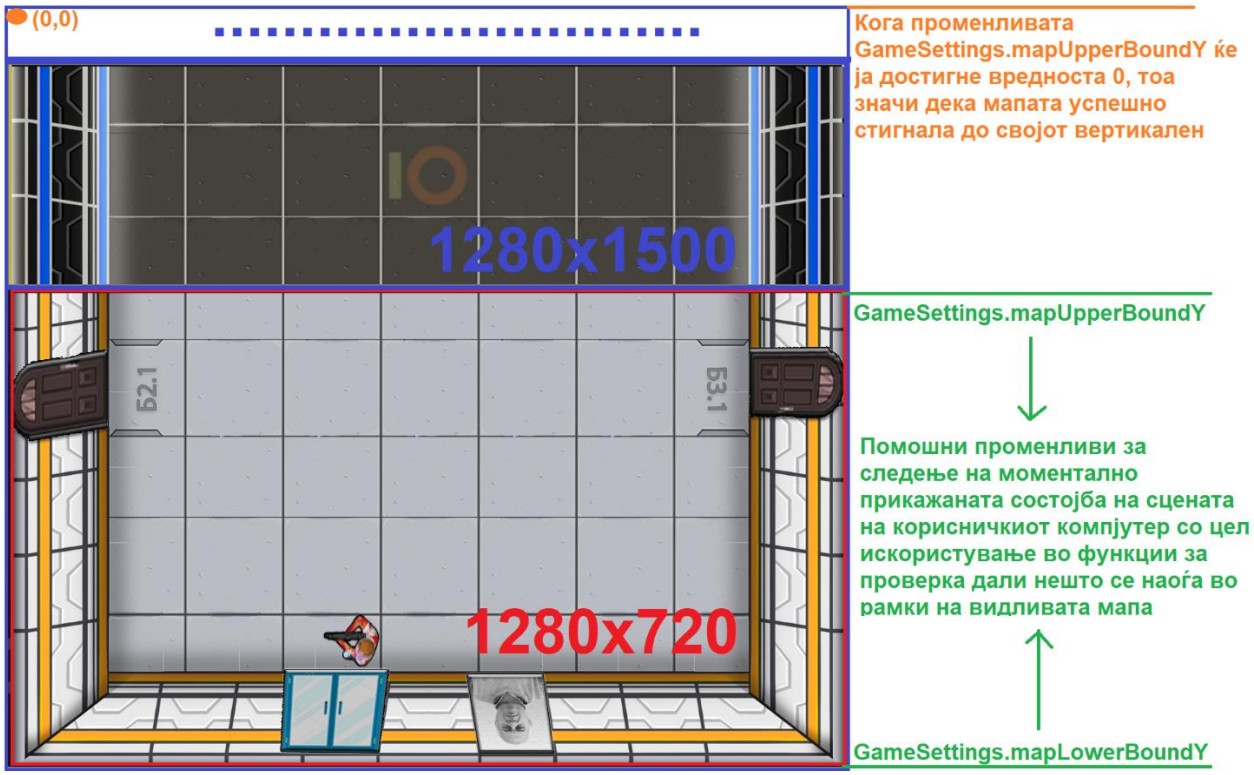
3.1. Генерална структура на проектот - класи, форми, контроли

Како инспирација и идеја за дизајнот на структурата на проектот за играта следевме примери изработени во C#, но и наши претходни проекти со кои сме работеле во истиот програмски јазик, но на други платформи. Структурата се состои од следните класи:

- **GameForm.cs –** главна форма која е еден вид визуелна рамка за играта која го зафаќа целиот екран со цел да му се овозможи на играчот да се внесе целосно во светот на играта без притоа да му пречат други елементи кои се дел од работната површина или отворените прозори на неговиот компјутер. Ги содржи главните информации за играта како: тековното ниво, поените и здравјето на играчот, како и неколку главни упатства за користење на апликацијата.
- **Models –** директориум во кој се содржани сите модели на ентитети присутни во апликацијата како: Player, Bullet, Book, Paper…
    - **VisualObject.cs -** апстрактна класа од која се изведуваат сите визуелни ентитети во апликацијато како што се играчот, непријателите и куршумите. Во неа се чуваат информации за позицијата на ентитетот во вид на координати, неговите висина и ширина, како и начинот на негова понатамошна визуелна репрезентација. Покрај ова, оваа класа содржи логика за проверка дали еден ентитет се наоѓа сеуште во рамките на мапата и дали се случило колизија со некој друг објект.
    - **Player.cs –** класа во која се наоѓаат основните атрибути на главниот играч: здравјето кое го поседува за време на студирањето, поените кои ги стекнува со уништување на секој противник, променливи кои ја определуваат неговата позиција на екранот, како и дополнителни карактеристики кои ја определуваат насоката и брзината на неговото движење по мапата.
    - **Bullet –** Класа за куршумите во играта, која ја содржи логиката за движење на куршумите и претставува потенцијална апстрактна класа од која можат да се изведат разни типови на куршуми кои би нанесувале различна штета на непријателите.
    - **Enemy.cs –** апстрактна класа која ги содржи заедничките карактеристика за непријателите (здравје, награда во вид на поени, брзина, штета која ја нанесуваат и сл.) во која се застапени сите типови на модели кои се наоѓаат во директориумот EnemyModels. Индивидуално, секој претставник кој наследува од оваа класа на свој начин го имплементира движењето и додава неколку посебни променливи и методи за својата класа.
    - **EnemyModels –** директориум во кој се наоѓаат моделите за непријателите кои наследуваат од апстрактната класа Enemy.cs, а тоа се ***Book.cs, Boss.cs и Paper.cs***.

- **Scene** – директориум во кој се наоѓа формата и логиката за визуелниот приказ на играта.
    - **GameView.cs** – Корисничка контрола *(User control)* која ја содржи формата со фиксни димензии (1280x720) каде всушност и се наоѓа целата игра. Содржи панел кој е со поголема висина од димензиите на матичната форма, со цел да се симулира вертикално придвижување на мапата. Во формата се справуваме со настани како кликање на тастатурата, вчитување на играта и сл. Истотака оваа форма го содржи главниот тајмер на апликацијата кој се извршува на интервал од 50ms и служи за справување со анимации и визуелно исцртување на играта и сите нејзини поделементи.
    - **Scene.cs** – главна класа во решението која што ја содржи поголемиот дел од логиката за функционалностите на играта. Служи како класа за комуникација меѓу моделите и визуелниот приказ. Содржи логика за креирање непријатели, куршуми, движење на мапата и слично. Во оваа класа се чуваат сите визуелни објекти кои се присутни во моментот кај корисникот со помош на податочната структура List.
    - **Animation** – класа во која е содржана логиката за анимирање на движењето на секој од визуелните објекти и се одвива со исцртување на множества од слики (*sprites*).
    - **AllAnimations** – класа во која се иницијализирани множеството од слики за секоја поединечна анимација на визуелните ентитети.
- **Settings –** директориум во кој се содржани поставките и константите поврзани со апликацијата, па така содржи две класи:
    - **Constants** – класа која ги содржи сите позначајни константи за играта како нивоа, дирекции на движење, случаен генератор и помошни координати за движење на главниот непријател за време на испитот.
    - **GameSettings –** класа која содржи променливи за големината на мапата, поставки дали играта да биде во DebugMode и дали мапата има ѕидови.

3.2. Имплементација на функционалноста: вертикален sidescroller



SideScroller функционалноста е имплементирана во GameView.cs корисничката контрола (*User Control*) чии димензии се 1280x720px каде е имплементиран панел со димензии 1280x1500 така што со континуирано придвижување на панелот вертикално од 0 до -780px се добива илузија на вертикално скролање. Ова може да се забележи во функцијата moveMap() која се повикува од sceneTimer функцијата (*објаснета подолу*) и го релоцира панелот вертикално за 2px секои 5ms и притоа користи променливи во кои ги чува информациите за тоа кој дел од панелот е прикажан на играчот (***mapLowerBoundY*** *и* ***mapUpperBoundY*** *од класата GameSettings*) кои се користат за неколку функции, меѓу кои и функцијата за детекција дали еден визуелен објект се наоѓа надвор од мапата.

```
public void moveMap()
{
    if (Map.Location.Y <= 0) // if it didn't reach the end
    {
        // Move the map by 2 px vertically
        Map.Location = new Point(Map.Location.X, Map.Location.Y + 2);
        GameSettings.mapLowerBoundY -= 2;
        GameSettings.mapUpperBoundY -= 2;
    }
}
```

```
public bool isInsideMap()
{
    // Object is considered in map if 50% of the height of the body is inside the map
    float objectVerticalBound = 0.5f * Height;
    int wallBounds = 0;
    if (GameSettings.mapHasWalls) wallBounds = GameSettings.wallBounds;

    if (PositionY > (GameSettings.mapUpperBoundY - objectVerticalBound))
    {
        if (PositionY < (GameSettings.mapLowerBoundY + objectVerticalBound))
        {
            .....
```

3.3. Опис на главната функција во апликацијата: sceneTimer_Tick(…)

Оваа функција се наоѓа во GameView.cs заедно со функцијата (***event handler***: *GameView_PreviewKeyDown()*) за преглед на сите притиснати копчиња во која се извршуваат функциите соодветни за притиснатото копче. Функцијата се извршува на секои 50ms и ги повикува сите потребни функции од сцената кои ја креираат целата логика на играта. Проверка за дали сме во играта или во почетното мени, па потоа извршување на движењето на мапата, куршумите и креирање и движење на непријателите. Истотака се извршува и логика за анимирањето на објектите, како и проверка за колизија или локацијата на играчот во случај да треба да се избрише некој објект кој бил или надвор од границите на мапата (за куршумите и играчот) или убиен (за непријателите).

```
107     void sceneTimer_Tick(object sender, EventArgs e)
108     {
109         this.Refresh();
110
111
112         if(isInGame)
113         {
114             gameScene.moveMap(); // Scroll the map vertically
115
116             gameScene.moveBullets(); // Move current bullets on map
117             gameScene.moveEnemies(); // Move the enemies if they are created
118             gameScene.createEnemies();
119
120             // Iterate through animation sprites
121             if (--animationTick <= 0)
122             {
123                 AllAnimations.nextImage();
124                 animationTick = 2;
125             }
126
127             // If the map has "eaten" the player or the player died
128             if (gameScene.player.PositionY > GameSettings.mapLowerBoundY || gameScene.player.Health <= 0)
129             {
130                 gameScene.resetLevel();
131                 openMenu();
132             }
133
134             //Collision detection removal of enemies
135             gameScene.enemies.RemoveAll(enemy => enemy.IsDead);
136             gameScene.activeBullets.RemoveAll(b => b.RemoveMark && b.Animation.isAnimFinished());
137         }
138     }
```