ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Δημιουργία δισδιάστατου βίντεο παιχνιδιού με χρηση αντικειμενοστραφούς C# στο πλαίσιο εργασίας XNA4

(2D video game development with object-oriented C# under XNA4 framework)

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: ΓΕΡΑΚΑΣ ΓΕΩΡΓΙΟΣ – ΗΛΙΑΣ

ΑΜ: 4929

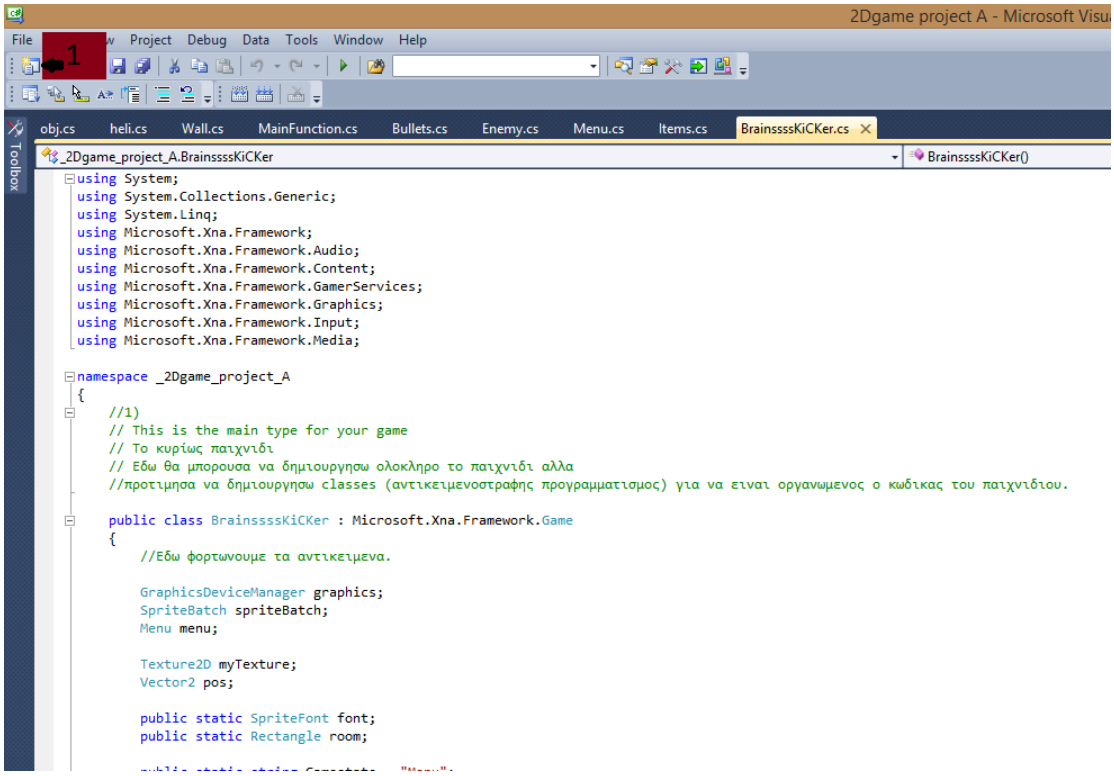
ΚΑΘΗΓΗΤΗΣ: ΚΩΝΣΤΑΝΤΑΡΑΣ ΑΝΤΩΝΙΟΣ

Χανιά, 20/10/2014

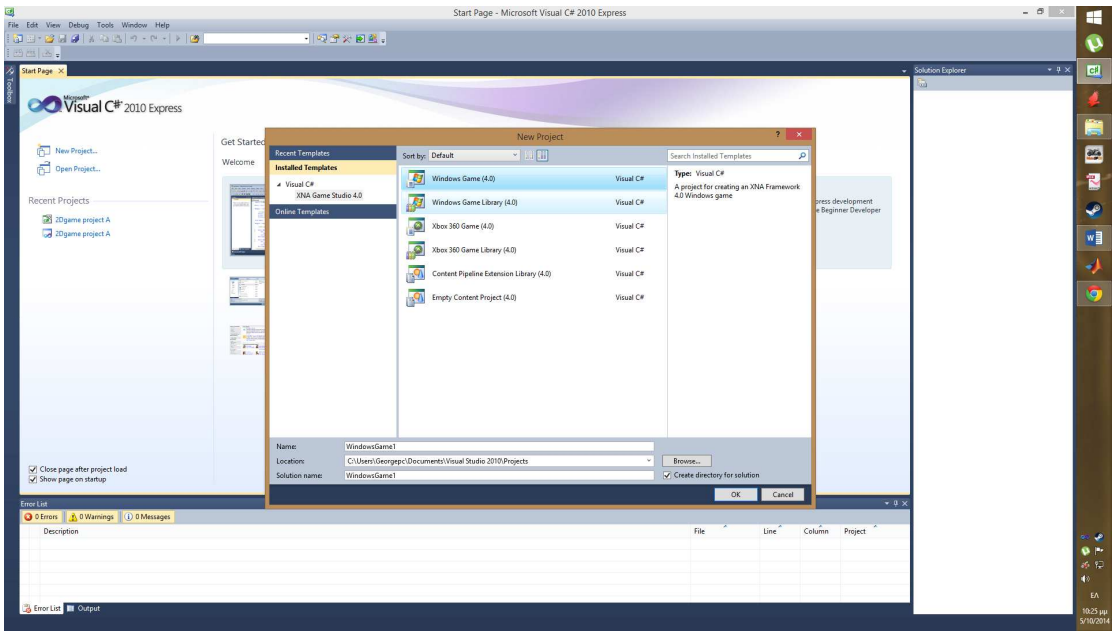
ΠΕΡΙΕΧΟΜΕΝΟ

Εξοικειωση με το προγραμμα Visual studio Express 2010

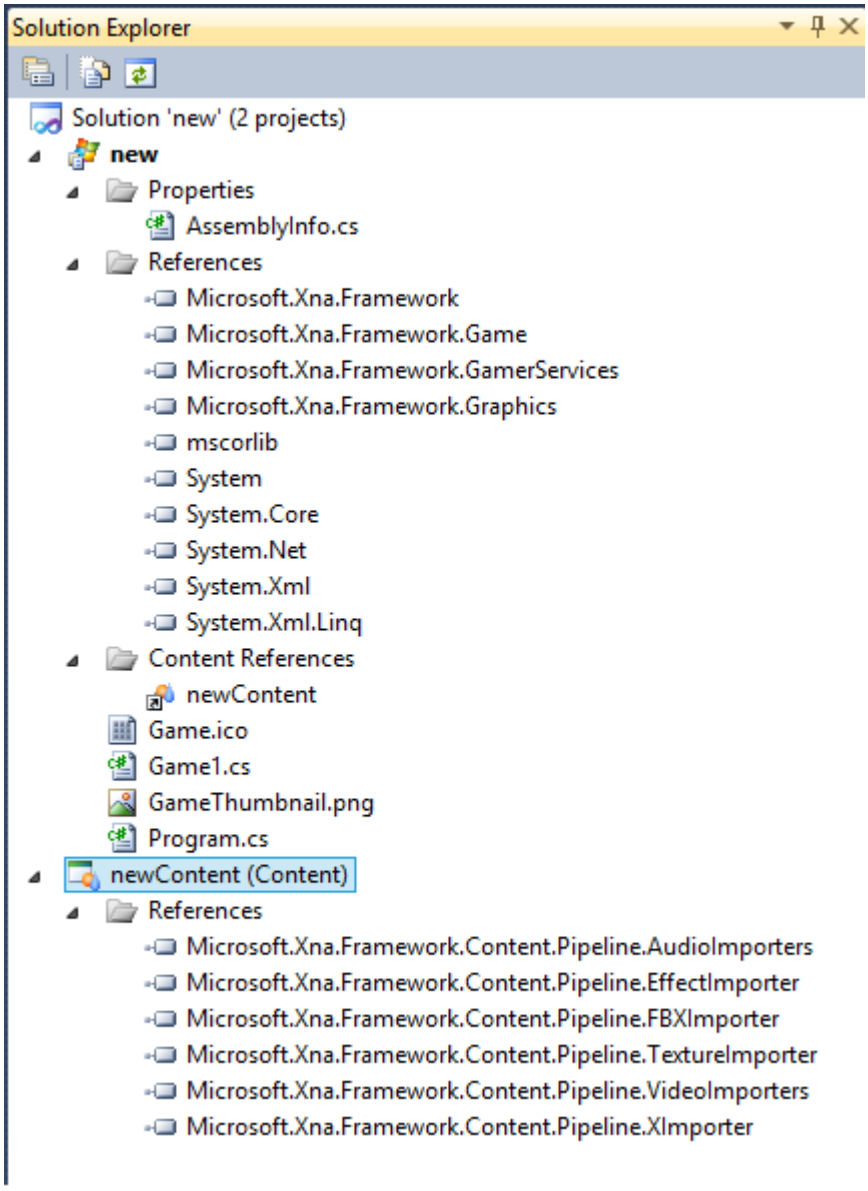
Στο σημείο 1 που φαίνεται στην εικόνα μπορούμε να δημιουργήσουμε ένα καινούριο project, όπως θα δούμε και στην συνέχεια. **[2]**



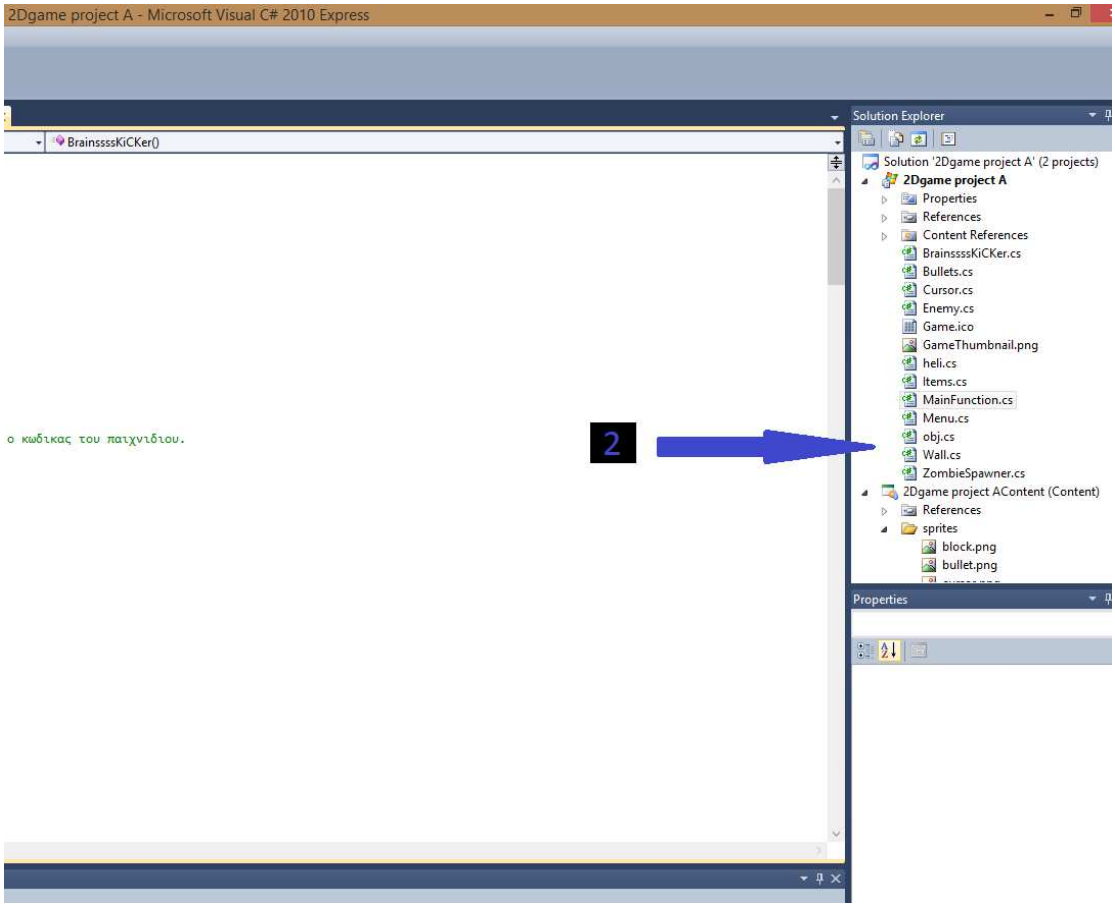
New project

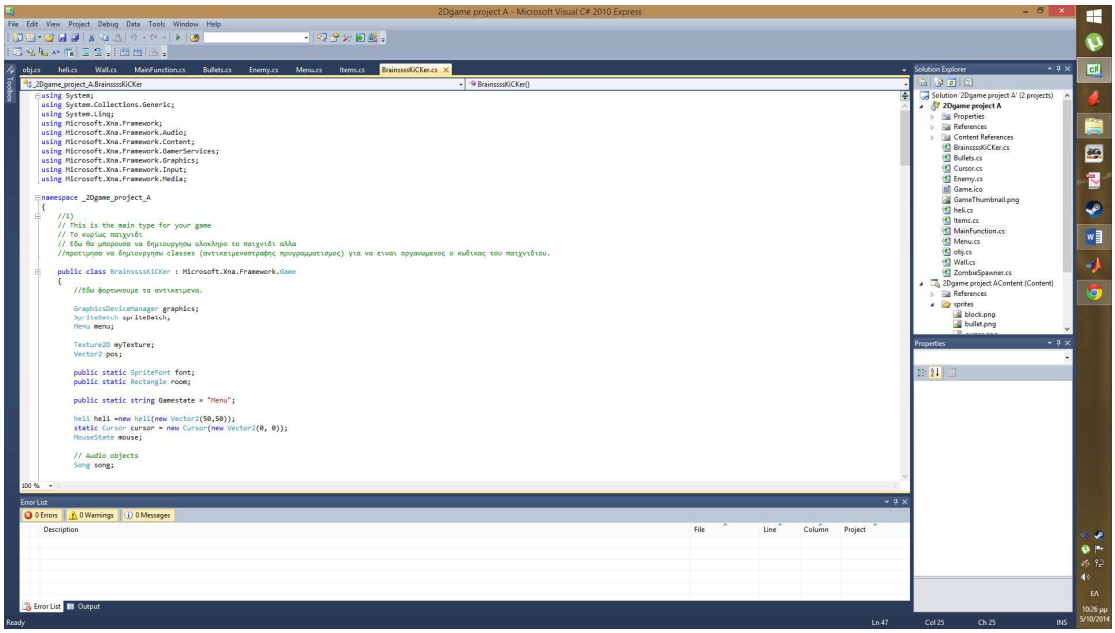


Τα παρακατω αρχεια δημιουργουνται αυτοματα απο την XNA οταν κανουμε βίντεο παιχνίδι για Windows ή XBOX. **[2]**



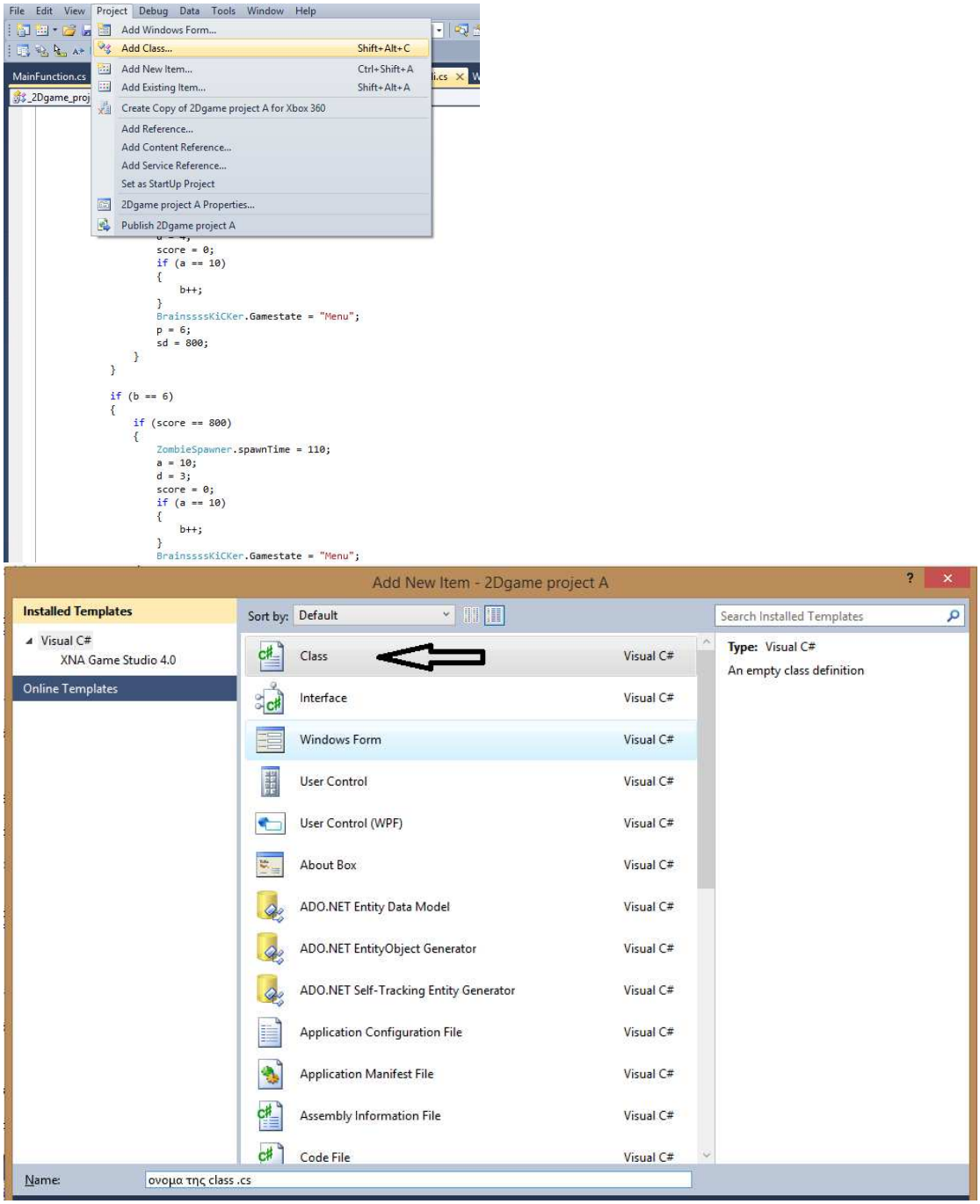
Στο σημείο 2 βρίσκονται όλες οι κλάσεις μας και τα αρχεία μας που έχουμε δημιουργήσει **[2]**





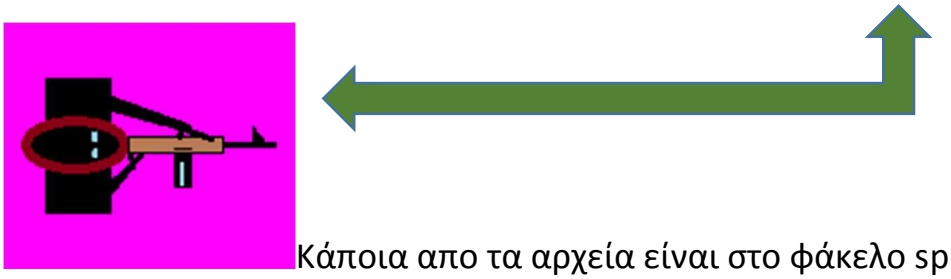
Με το πλήκτρο F5 τρέχει το πρόγραμμα(Debug)

Δημιουργια class**[2]**

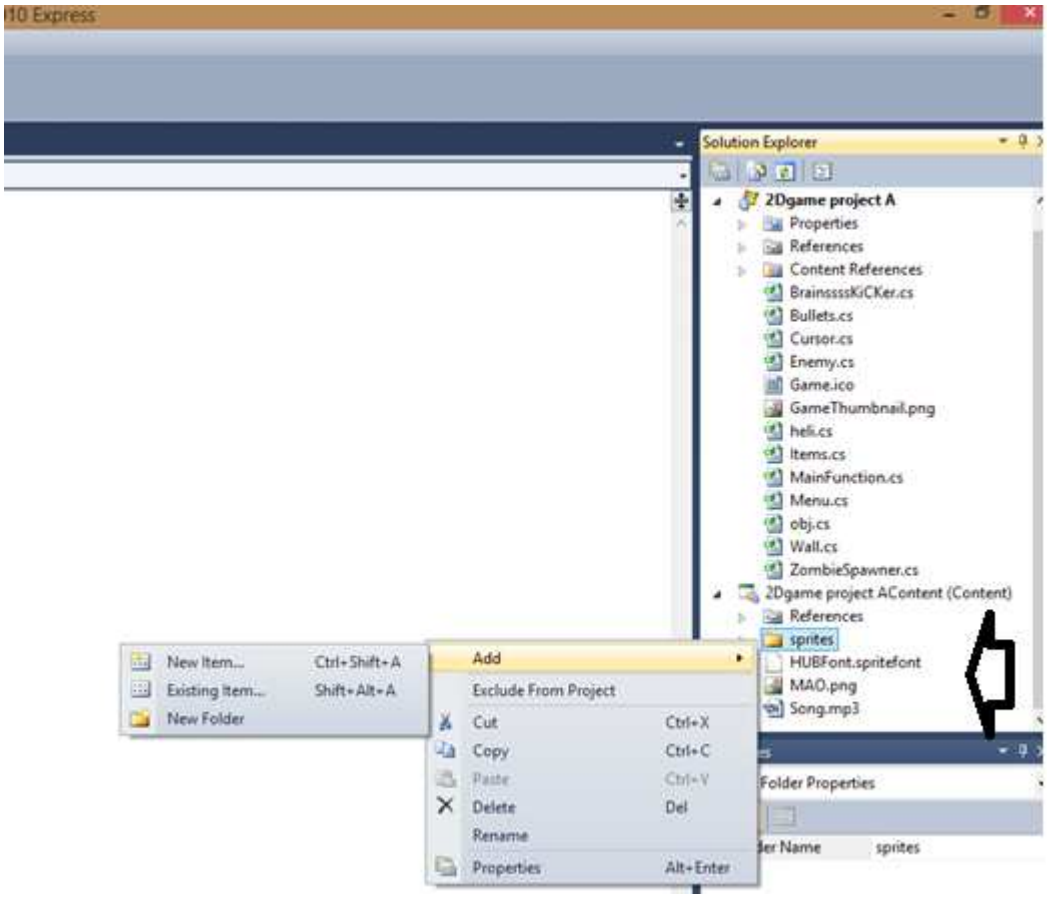


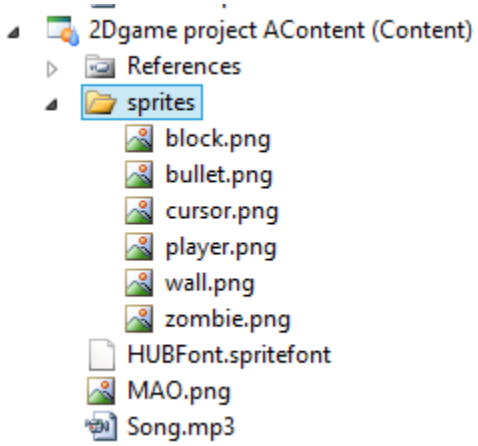
Magenta είναι ένα χρωμα που δεν μπορεί να φανεί στην οθόνη και το βρίσκεις με κωδικό: RGB: 255, 0, 255 ή HSL: 0.83, 1.00, 0.50 έτσι δεν φαίνεται το ρόζ πλαίσιο(Magenta) πίσω απο τα αντικείμενα. **[1]**

Εισαγωγή αρχείων γραφικών όπως το παρακάτω αρχείο (png).



Κάποια απο τα αρχεία είναι στο φάκελο sprites. **[2]**





Σύντομη εισαγωγη στη C# [11]

Η C# είναι μία ολοκληρωμένη αντικειμενοστραφής γλώσσα προγραμματισμού σχεδιασμένη για τη δημιουργία λογισμικού σε .Net Framework.

- Τα πάντα στη C# είναι αντικείμενα
- Παρέχει άμεση πρόσβαση σε τεράστιες βιβλιοθήκες κλάσεων του .Net Framework & ασφάλεια των τύπων της.

Θα δούμε παρακάτω τη σύνταξη και τη δομή ενός απλού προγράμματος (Console Application) στη C# που εμφανίζει το μήνυμα ‘’Hello World’’ και θα αναλύσουμε γραμμή γραμμή τα μέρη του:

```
1.    using System;

2.    namespace MyNamespace(ονομα του project)
3.
4.    {
5.       class Helloworld
6.       {
7.        static void Main()
          {
            Console.WriteLine("Hello World");
            Console.ReadKey();

          }
        }
      }
```

Σε ένα C# πρόγραμμα χρησιμοποιούμε Namespaces για την οργάνωση του κώδικα μας. **[11]**

Στο παραπάνω πρόγραμμα χρησιμοποιήσαμε την βιβλιοθήκη System (γραμμή 1 του παραδείγματος) μέσο της λέξης κλειδιού using ,το οποίο περιλαμβάνει έτοιμες κλάσεις και μεθόδους του .NET με έτοιμο επαναχρησιμοποιήσιμο κώδικα(Συναρτήσεις).

Ένα Namespace μπορεί να περιέχει τους εξής τύπους :

- Namespaces(ονομα του project)
- Classes(Ετικέτες για να διαχειρίζονται τα αντικείμενα)
- Delegates(Συναρτήσεις)
- Enums(Λίστες)
- Structs(δομές)
- Interfaces(Διασυνδέσεις)

Για τη δημιουργία ενός δικού μας Namespace χρησιμοποιούμε τη δεσμευμένη λέξη-κλειδί namespace (γραμμή 2), έτσι η κλάση Helloworld (γραμμή 4) εμπεριέχεται στο MyNamespace που δημιουργήσαμε.

Στη συνέχεια του κώδικα μπορούμε να δούμε πως μπορούμε να εμφανίσουμε στη γραμμή εντολών το μήνυμα "Hello World". Η μέθοδος Main() (γραμμή 6) περιέχει την εντολή Console.WriteLine("Hello World"); ,η οποία εμφανίζει το τελικό μήνυμα.

Η κλάση Console ανήκει στο χώρο ονομάτων System
και η WriteLine() είναι μέθοδος της Console.
Εάν δεν χρησιμοποιούσαμε την εντολή using System; Η εντολή θα συντασσόταν System.Console.WriteLine("Hello Word");
Χρησιμοποιούμε την τελεία ανάμεσα από Namespaces, κλάσεις και μεθόδους για να δείξουμε τι εμπεριέχεται σε τι. Πχ.

Τα σχόλια συντάσσονται χρησιμοποιώντας αυτόν τον συμβολισμό (//) ,εάν επρόκειτο για μία γραμμή σχολίου.
//Αυτό είναι ένα σχόλιο

Εισαγωγη στη μηχανές παιχνιδιού[1]

Μια **μηχανή παιχνιδιού** είναι ένα σύστημα λογισμικού σχεδιασμένο για τη δημιουργία και την ανάπτυξη βιντεο παιχνιδιών.

Υπάρχουν πολλές μηχανές παιχνιδιών οι οποίες είναι σχεδιασμένες να δουλεύουν σε κονσόλες βιντεοπαιχνιδιών και λειτουργικά συστήματα επιτραπέζιων υπολογιστών όπως τα Microsoft Windows, το Linux, και το Mac OS X.

Η κεντρική λειτουργικότητα που παρέχεται τυπικά από μια μηχανή παιχνιδιού περιλαμβάνει μια μηχανή φωτοαπόδοσης ("renderer") για 2D ή 3D γραφικά, μια μηχανή φυσικής ή εντοπισμού συγκρούσεων(collision detection, καθώς και collision response), ήχο, Scripting(ενα αντικείμενο οταν δέχεται ένα ερέθισμα αλλάζει την κατάστασή του), Animation(οτιδήποτε δεν ειναι ζωντανο, ‘φανταστικο’ και δεν ειναι απαραιτητο να υπακουει στους νόμους της φυσικής), τεχνητή νοημοσύνη, δικτύωση, Streaming(παιχνίδι σε πραγματικό χρόνο μέσω ίντερνετ), διαχείριση μνήμης, νήματα (threading), υποστήριξη τοπικοποίησης, και ένα γράφο σκηνής (scene graph).

Η διαδικασία της ανάπτυξης παιχνιδιού συχνά οικονομικοποιείται με το ότι σε μεγάλο μέρος η ίδια μηχανή παιχνιδιού επαναχρησιμοποιείται για να δημιουργηθούν διαφορετικά παιχνίδια.

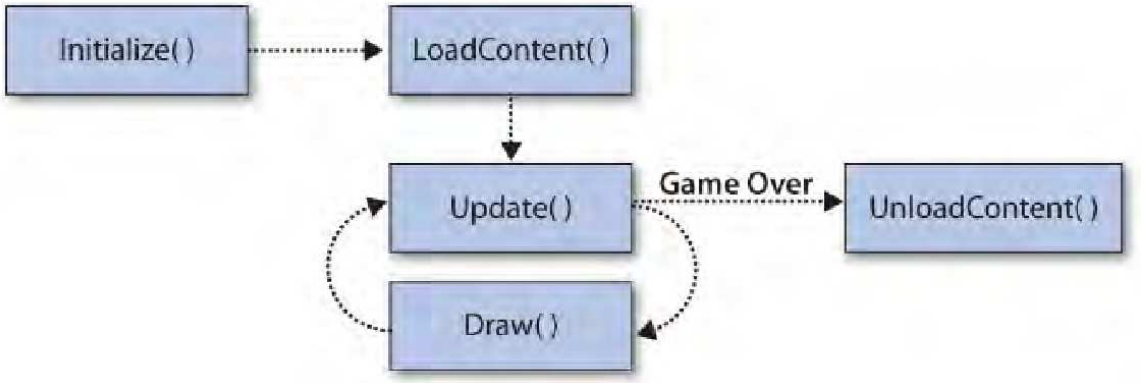
Εισαγωγή στη XNA 4.0 (X.N.A. δεν ειναι ακρώνυμα, το X σημαινει σε κώδικα Μορς "-..-" , και το NA σημαινει "-. .-") [1]

Η Microsoft XNA είναι ένα σύνολο εργαλείων με ένα διαχειριζόμενο περιβάλλον εκτέλεσης που διευκολύνει τη δημιουργία βίντεο παιχνιδίων.Η XNA τρέχει σε windows,windows phone, XBOX και άλλες πλατφόρμες της Microsoft διότι βασίζεται στο πλαίσιο .NET της Microsoft.

Game1.cs*

WindowsGame2.Game1

```
public class Game1 : Microsoft.Xna.Framework.Game
{
    GraphicsDeviceManager graphics;
    SpriteBatch spriteBatch;
    public Game1()
    {
        graphics = new GraphicsDeviceManager(this);
        Content.RootDirectory = "Content";
    }
    protected override void Initialize()
    {
        // TODO: Add your initialization logic here

        base.Initialize();
    }
    protected override void LoadContent()
    {
        // Create a new SpriteBatch, which can be used to draw textures.
        spriteBatch = new SpriteBatch(GraphicsDevice);

        // TODO: use this.Content to load your game content here
    }
    protected override void UnloadContent()
    {
        // TODO: Unload any non ContentManager content here
    }
    protected override void Update(GameTime gameTime)
    {
        // Allows the game to exit
        if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
            this.Exit();

        base.Update(gameTime);
    }
    protected override void Draw(GameTime gameTime)
    {
        GraphicsDevice.Clear(Color.CornflowerBlue);

        base.Draw(gameTime);
    }
}
}
```



Βασική δομή της XNA 4.0: [8]

Το αρχείο **namespace** Το όνομα του project() που περιέχει αρχικοποιήσεις, το graphics device(συσκευη γραφικων: δίνει στο παιχνίδι πρόσβαση στο υλικό του υπολογιστό).Οι παρακάτω συναρτήσεις είναι απαραίτητες.

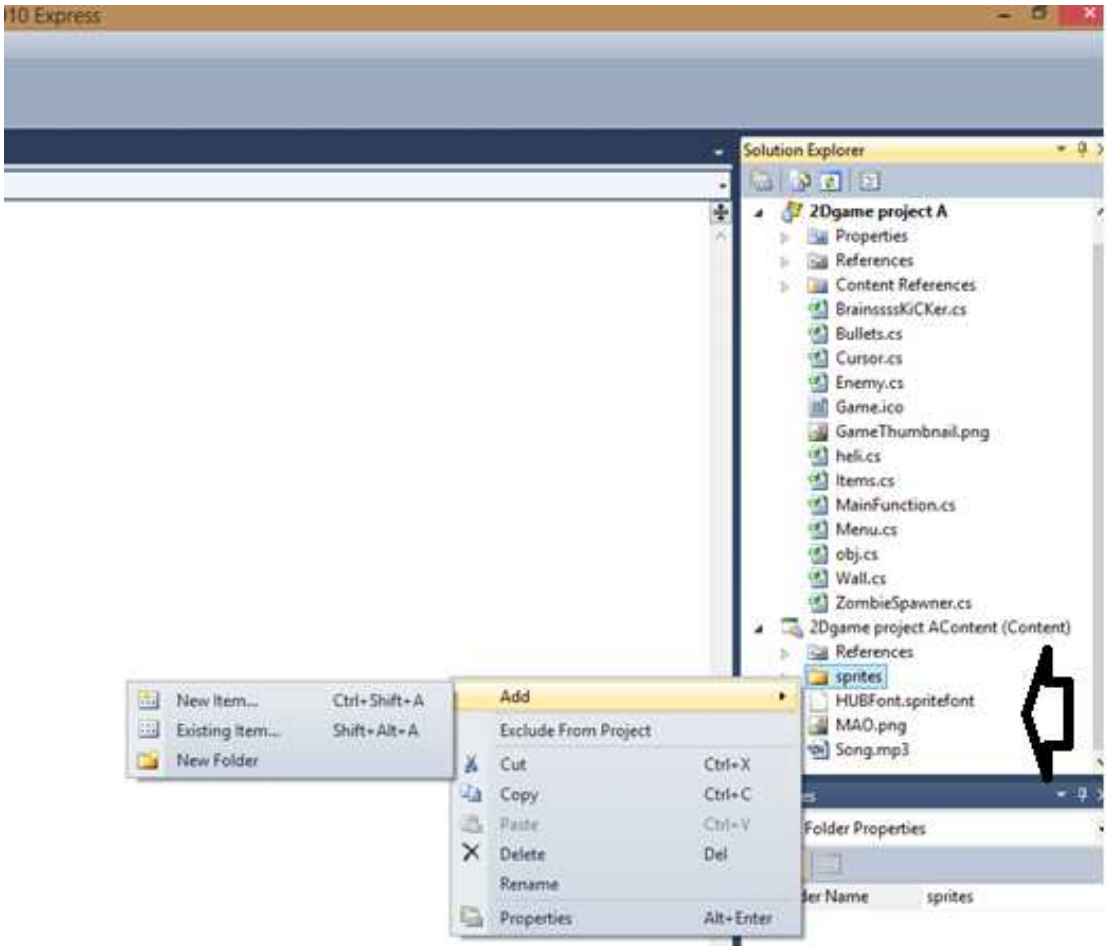
- Το όνομα του παιχνιδιού() Main class.
- Initialize() Προετοιμάζει το περιβάλλον του παιχνιδιού π.χ. ορίζει την αρχική κατάσταση των αντικειμένων.
- LoadContent() Φορτώνει τα αρχεία που χρειάζεται το παιχνίδι π.χ μουσική,φώτογραφιες σε μορφή (.png).
- Update () Εδώ γίνεται ο έλεγχος των εισόδων του χρήστη (ποντίκι, πληκτρολόγιο) που επηρεάζουν την αντίδραση των αντικειμένων. Ακόμα εδώ περιέχονται ολες οι συναρτήσεις που κάνουν τα αντικείμενα δυναμικά(να κινούνται ή οτι άλλο είναι προγραμματισμένα να κάνουν).
- Draw()Αυτή η συνάρτηση ειναι υπεύθυνη για την εμφάνιση των αντικειμένων στο περιβάλλον του παιχνιδιού.
- Αξίζει να σημειωθεί οτι η συνάρτηση **Update** και **Draw** εκτελούνται 60 φορές το δευτερόλεπτο.
- UnloadContent()Εκφορτώνει τους πόρους του παιχνιδιού.

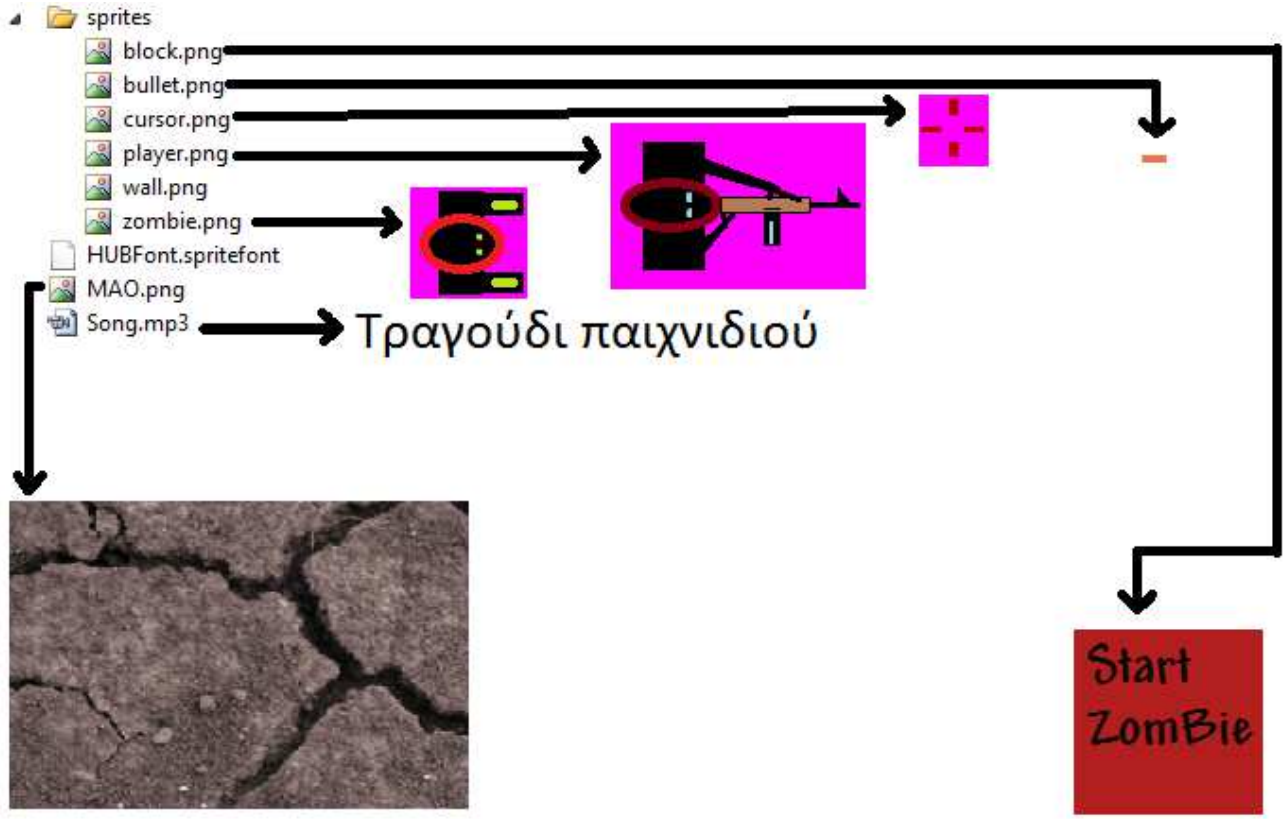
Sprites(Γραφικά ή Εικονες) [8]

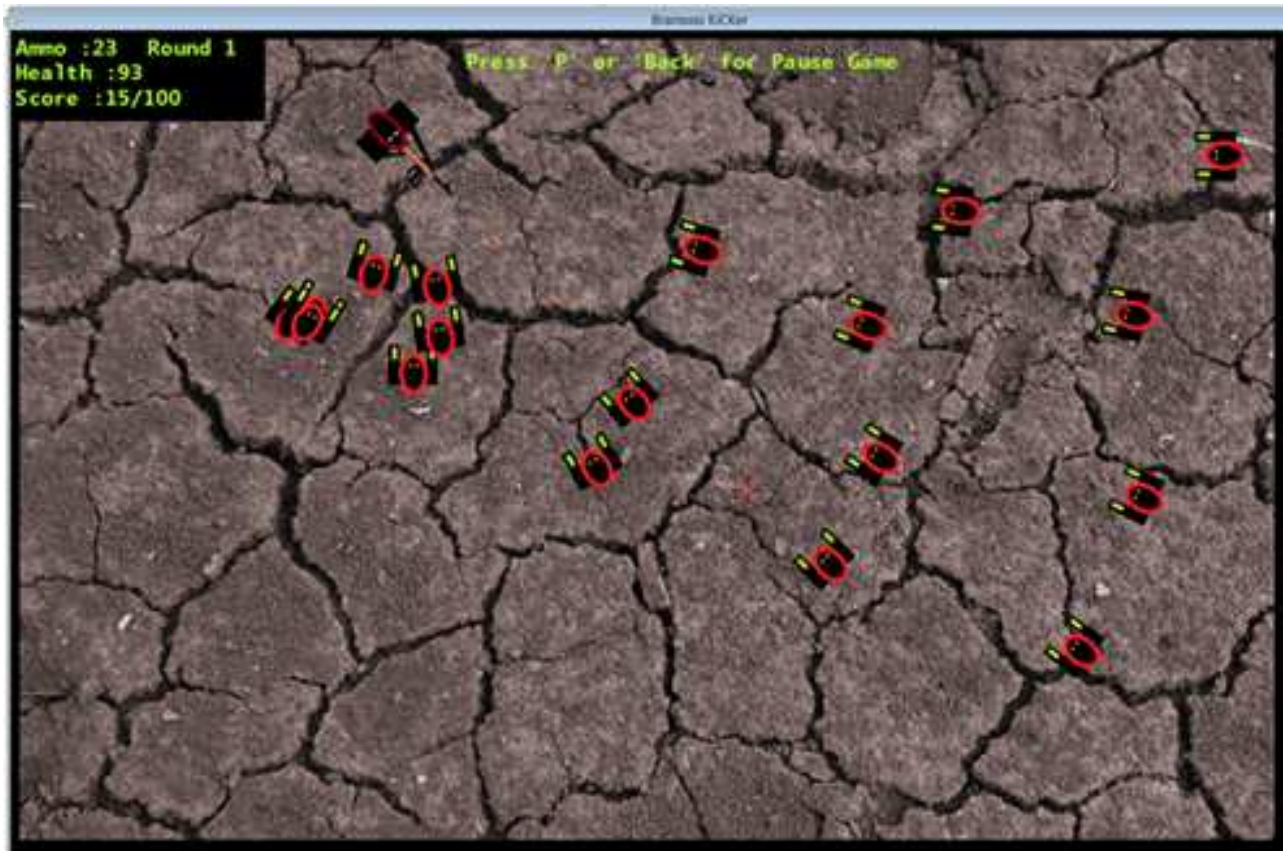
Sprites ειναι δισδιάστατες εικόνες που ελέγχονται απο το κώδικα κατά τη διάρκεια του παιχνιδιού. Χρησιμοποιούνται για να ζωγραφιστεί το αντικείμενο στο παιχνίδι.

Spritebatch είναι μια λίστα απο εικόνες(sprites).

Spritefont είναι ένα αρχείο που χρησιμοποιείται για να ορίσει τον τύπο της γραμματοσειράς που θα εμφανιστεί στο παιχνίδι.Το αρχείο Spritefont τοποθετείται στο 2Dgame project AContent(π.χ. στην επόμενη σελίδα τα μηνύματα που εμφανίζονται στο παιχνίδι με πράσινο χρώμα).




sprites
block.png
bullet.png
cursor.png
player.png
wall.png
zombie.png
HUBFont.spritefont
MAO.png
Song.mp3
Τραγούδι παιχνιδιού
Start
ZomBie


Ammo :23 Round 1
Health :93
Score :15/100
Press P or Back for Pause Game

Διανύσματα & Διαστάσεις(Vector2 ,Rectangle) [8]

Για φορτώσουμε εικόνες με το Spritebatch πρέπει να δημιουργηθεί ενα περιβάλλον (Texture2D). Στην XNA ο χώρος μπορεί να είναι δισδιάστατος ή τρισδιάστατος. Στη συγκεκριμένη εργασία είναι δισδιάστατος.

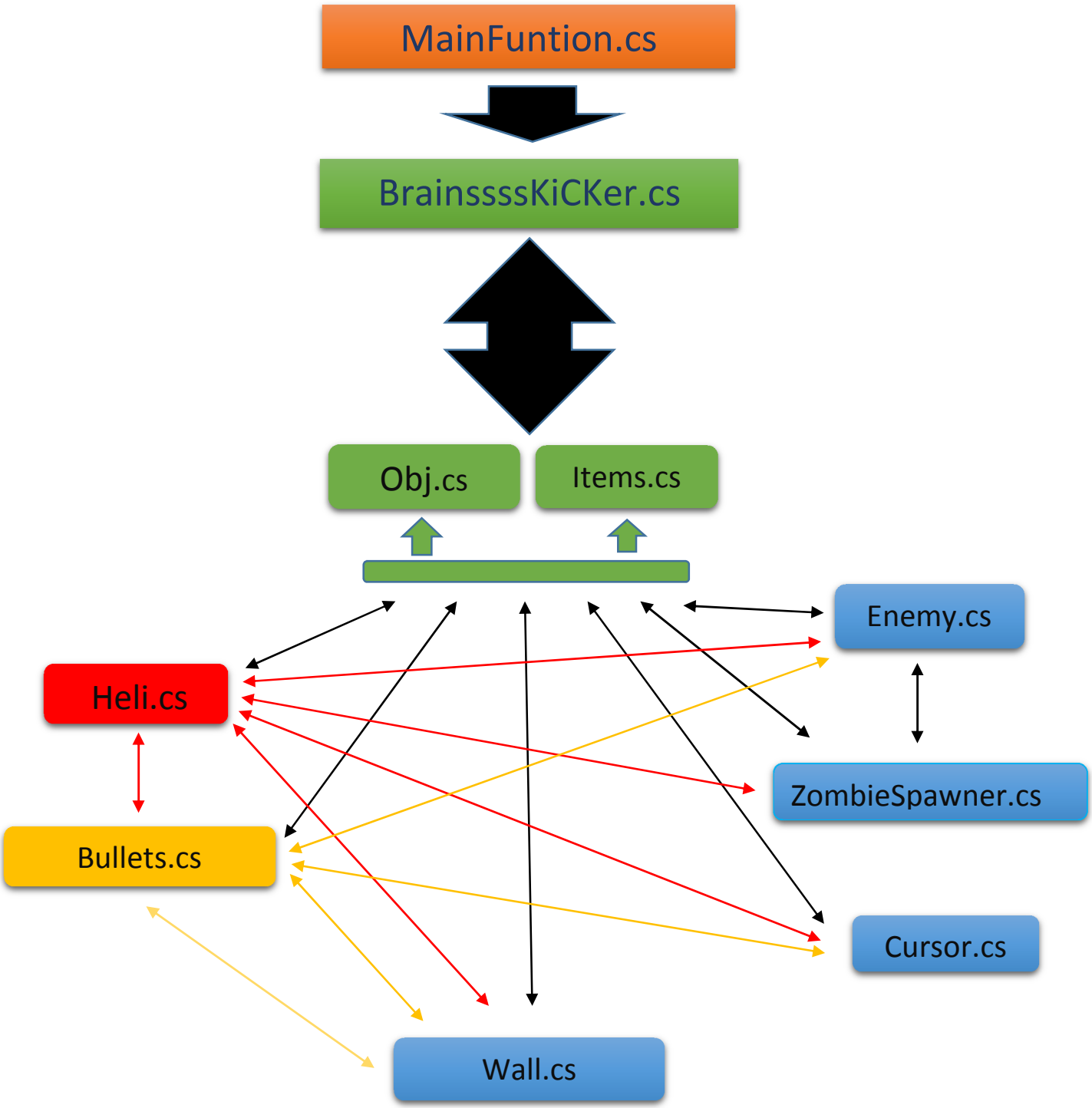
```
myTexture = Content.Load<Texture2D>("MAO");//MAO είναι το
ονομα της φωτογραφιας του backround.
```

Για να μπορούμε να φορτώσουμε εικόνες με το Spritebatch χρειάζεται να έχουμε δηλώσει την αρχική θέση του αντικειμένου, αυτο γίνεται με τη συνάρτηση Vector2(x,y) (2D, παίρνει δύο ορίσματα).

Το μέγεθος της εικόνας ρυθμίζεται απο τη συνάρτηση Rectangle().

```
//φορτωνουμε το περιεχομενο των αντικειμενων.
protected override void LoadContent()
{
    // Create a new SpriteBatch, which can be used to draw textures.
    spriteBatch = new SpriteBatch(GraphicsDevice);

    font = Content.Load<SpriteFont>("HUBFont");

    menu.LoadContent(Content);

    myTexture = Content.Load<Texture2D>("MAO");//Φορτωνει το backround.
    pos = new Vector2(10, 10);
    foreach (obj o in Items.objList)
    {
        o.LoadContent(this.Content);
    }

    //φορτωνουμε μουσικη.

    song = Content.Load<Song>("song");
```

ΔΙΑΓΡΑΜΜΑ ΡΟΗΣ



ΑΝΑΛΥΣΗ ΚΩΔΙΚΑ ΤΟΥ ΠΑΙΧΝΙΔΙΟΥ

Κυριως προγραμμα: [3]

```
using System;

namespace _2Dgame_project_A
{
#if WINDOWS || XBOX
   //Αυτο το Class ειναι δημιουργειται αυτοματα απο το XNA, χρειαζεται για να ...
   //... ξεκινησει το παιχνιδι και δεν χρειαζεται επεξεργασια.
   static class MainFunction
   {
      static void Main(string[] args)
      {   // BrainssssKiCKer ειναι το κεντρικο παιχνιδι
         using (BrainssssKiCKer game = new BrainssssKiCKer())
         {
            game.Run();// εντολη για να ξεκινησει το προγραμμα(πιο αναλυτικα αυτο το
προγραμμα ...   //...υπαρχει απο την xna απλα για να ξεκινησει την BrainssssKiCKer ταξη).
         }
      }
   }
#endif
}
```

Αντικειμενο: BrainssssKiCKer.cs(Κυρίως Κώδικας)

```
//Απαιτουμενες βιβλιοθηκες[3]
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

namespace _2Dgame_project_A
{
   // Το κυρίως παιχνιδι
   // Εδω θα μπορουσα να δημιουργησω ολοκληρο το παιχνιδι αλλα
   //προτιμησα να δημιουργησω classes (αντικειμενοστραφης προγραμματισμος) για να
ειναι οργανωμενος ο κωδικας του παιχνιδιου ως προς τα αντικειμενα. Ετσι καλουμε εδω
ολα τα υπολοιπα αντικειμενα(Menu, obj, Heli, Enemy, Cursor, Wall, Bullets, Items,
ZombieSpawner)
)

   public class BrainssssKiCKer : Microsoft.Xna.Framework.Game
   {
      //ΑΡΧΙΚΟΠΟΙΗΣΕΙΣ.

      GraphicsDeviceManager graphics; //Οριζεται ως Διαχειριστης γραφικων της XNA.
      SpriteBatch spriteBatch; //
      Menu menu; //Αρχικοποιηση του μενου

      Texture2D myTexture; //αρχικοποιηση μιας μεταβλητης(εικονας) 2 διαστασεων.
      Vector2 pos; // αρχικοποιηση της θεσης 2 διαστασεων

      public static SpriteFont font; // αρχικοποιηση της μεταβλητης ΛΕΞΕΩΝ.

      public static string Gamestate = "Menu";//

      static heli heli =new heli(new Vector2(150,150)); //Καλουμε τη ταξη του παικτη.
      static Cursor cursor = new Cursor(new Vector2(50, 50)); //Καλουμε τη ταξη του στοχου.
      MouseState mouse; // αρχικοποιηση της καταστασης του ποντικιου.


      Song song; // αρχικοποιηση μιας μεταβλητης μουσικης.

      public static Rectangle screen; // αρχικοποιηση της μεταβλητης ΜΕΓΕΘΟΥΣ.
      public static Rectangle room; //   αρχικοποιηση της μεταβλητης ΜΕΓΕΘΟΥΣ.
```

```
        public BrainssssKiCKer()//Βασικο αντικειμενο.
        {
            //Ρυθμιση της αναλυσης της οθονης
            graphics = new GraphicsDeviceManager(this);
            graphics.PreferredBackBufferWidth = 1600; //(διασταση του πλατους του
'παραθυρου' του παιχνιδιου)
            graphics.PreferredBackBufferHeight = 1000; //(διασταση του του υψους του
'παραθυρου' του παιχνιδιου)

            graphics.ApplyChanges();//Η εντολη αυτη εφαρμοζει τη παραπανω αναλυση οθονης.


            Content.RootDirectory = "Content";// Η διευθυνση των περιεχομενων της XNA.
        }

       //Η συναρτηση Initialize()  προετοιμάζει το περιβάλλον του παιχνιδιού π.χ. ορίζει την
αρχική  κατάσταση των αντικειμένων

        protected override void Initialize()[6]
        {
            Items.Initilize();
            room = new Rectangle(0, 0, graphics.PreferredBackBufferWidth * 4,
graphics.PreferredBackBufferHeight * 4); //Εκχωρει το μεγεθος της οθονης*4

            screen = new Rectangle(0, 0, graphics.PreferredBackBufferWidth,
graphics.PreferredBackBufferHeight); // Εκχωρει το μεγεθος της οθονης.

            menu = new Menu();//Καλειται το αντικειμενο μενου.

            base.Initialize();//Αρχικοποιηση της βασης δεδομενων.
        }

         //φορτωνουμε τα αρχεια των αντικειμενων.
        protected override void LoadContent()
        {
            // Δημιουργουμε ενα καινουργιο SpriteBatch,η χρηση του ειναι για την εμφανισει
//γραφικων.
            spriteBatch = new SpriteBatch(GraphicsDevice);

            font = Content.Load<SpriteFont>("Grammatoseira");//φορτωνει το αρχειο

            menu.LoadContent(Content); //φορτωνει το αρχειο του  μενου

            myTexture = Content.Load<Texture2D>("MAO");//φορτωνει το αρχειο του
                                                       //background.

            pos = new Vector2(10, 10); //Αρχικοποιει τη θεση των αντικειμενων.

            foreach (obj MYobject in Items.objList)
            {//Για οσα αντικειμενα περιεχονται στη λιστα να φορτωνονται οι εικονες τους.
                MYobject.LoadContent(this.Content);
```

```
            }
            song = Content.Load<Song>("song"); //φορτωνουμε μουσικη.

            MediaPlayer.Play(song); //Παιζει η μουσικη.

            MediaPlayer.IsRepeating = true; //Η μουσικη μολις τελειωσει επαναλαμβανεται.

             cursor.LoadContent(Content); //φορτωνει το αρχειο του κερσορα.
        }

protected override void UnloadContent()
        {
          //Εκφορτώνει τους πόρους του παιχνιδιού.
        }
       // Εδώ γίνεται ο έλεγχος των εισόδων του χρήστη (ποντίκι, πληκτρολόγιο) που
επηρεάζουν την   αντίδραση των αντικειμένων. Ακόμα εδώ περιέχονται ολες οι συναρτήσεις
που κάνουν τα    αντικείμενα δυναμικά(να κινούνται ή οτι άλλο είναι προγραμματισμένα να
κάνουν)
        protected override void Update(GameTime gameTime) [6]
        {
          //Η συνάρτηση Update εκτελείται 60 φορές το δευτερόλεπτο.

          if (Gamestate =="Exit" )  // Η εντολη αυτη κλεινει το παιχνιδι απο το κουμπι του
μενου EXIT.
             this.Exit();

          Window.Title = "Brainssss KiCKer";//Ο τιτλος πανω στο παραθυρο του παιχνιδιου.
```

Η εντολη switch (Gamestate) δομειτε οπως στη εικονα παρακατω

```
switch (caseSwitch)
{
    // The following switch section causes an error.
    case 1:
        Console.WriteLine("Case 1...");
        // Add a break or other jump statement here.
    case 2:
        Console.WriteLine("... and/or Case 2");
        break;
}
```

```
switch (Gamestate) //Εντολη για την αλλαγη της καταστασης του παιχνιδιου.

{
  case "Game":
    foreach (obj MYobject in Items.objList)
    {//Ενημερωνεται καθε αντικειμενο της λιστας MYobject.
      MYobject.Update();
    }
    break;
  case "Menu":
    menu.Update(gameTime); //Ενημερωνεται το αντικειμενο Menu.
    break;
}
cursor.Update();//Ενημερωση της καταστασης του ποντικιου.
mouse = Mouse.GetState();//θετει στη μεταβλητη τη καταστση του ποντικιου.

base.Update(gameTime); //Ενημερωση της βασης δεδομενων.
}
```

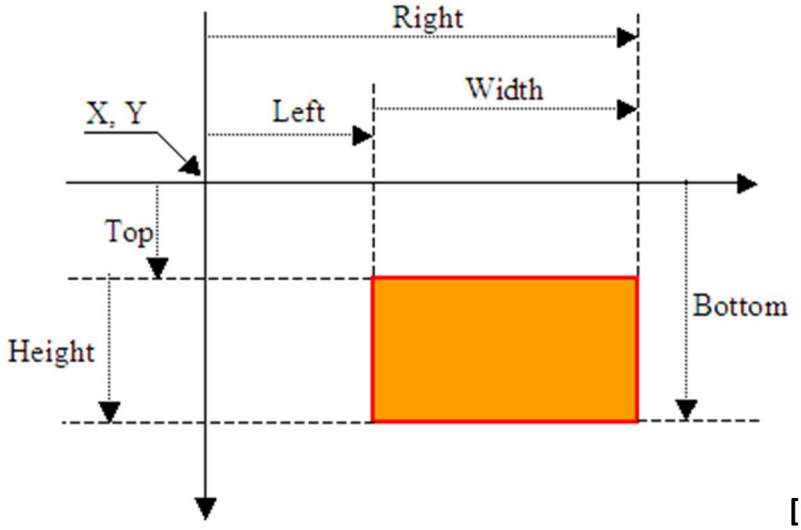
```
// Εδω ελεγχουμε τι θα εμφανιστει το περιβαλλον του παιχνιδιου
    protected override void Draw(GameTime gameTime) [6]
    {
      //Η συνάρτηση Draw εκτελείται 60 φορές το δευτερόλεπτο.
      //Το χρωμα που θα εχει το φόντο του παιχνιδιου.
      GraphicsDevice.Clear(Color.Black);
      switch (Gamestate)  //Εντολη για την αλλαγη της καταστασης του παιχνιδιου.
      {
        case "Game":
          spriteBatch.Begin();
          //Στην XNA για να εμφανιστουν καποια γραφικα στο παιχνιδι με τη συναρτηση
        draw ειναι απαραιτητο ο κωδικας για την φωτογραφια να μπει αναμεσα στις
        εντολες spriteBatch.Begin() και spriteBatch.End().[5]
```

```
            spriteBatch.Draw(myTexture, pos, Color.White); //Εμφανιζεται φωτογραφια
'Background'. [6]
            foreach (obj MYobject in Items.objList)
            {//Εμφανίζεται καθε αντικειμενο της λιστας MYobject.
               MYobject.Draw(spriteBatch);
            }
            cursor.Draw(spriteBatch); // Εμφανίζεται ο κερσορας(στοχος).
            spriteBatch.End();

            if (Keyboard.GetState().IsKeyDown(Keys.F1))
            {
               graphics.IsFullScreen = true; // Εφαρμογη πληρης οθονη οταν πατιεται το F1.
            }
            graphics.ApplyChanges();
            if (Keyboard.GetState().IsKeyDown(Keys.F2))
            {
               graphics.IsFullScreen = false; // Εφαρμογη 'παραθυρου' οταν πατιεται το F2.
            }
            graphics.ApplyChanges();//Αλλαζει τις ρυθμισεις στις παραπανω.

            break;
         case "Menu" :
            menu.Draw(spriteBatch);
            // Εφαρμογη του παιχνιδιου σε πληρη οθονη οταν πατιεται το F1.
            if (Keyboard.GetState().IsKeyDown(Keys.F1))
            {
               graphics.IsFullScreen = true;
            }
            graphics.ApplyChanges();
            // Εφαρμογη του παιχνιδιου σε 'παραθυρο' οταν πατιεται το F1.
            if (Keyboard.GetState().IsKeyDown(Keys.F2))
            {
               graphics.IsFullScreen = false;
            }
            graphics.ApplyChanges();//Αλλαζει τις ρυθμισεις στις παραπανω.

          break;
        }
        base.Draw(gameTime); //Επιστρεφει στη βαση δεδομενων το gametime.
    }
    public static Vector2 GetCursorPos()
    {
        return cursor.position; //επιστρεφει τη θεση του κερσορα.
    }
  }
}
```

ΑΝΤΙΚΕΙΜΕΝΟ:Obj.cs

```
//Απαιτουμενες βιβλιοθηκες[3]
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

namespace _2Dgame_project_A
{
   class obj[6]
   {
          //Η ταξη(class) αυτη ,ειναι υπευθυνη για την δημιουργια των αντικειμενων  του ...
          //... παιχνιδιου ,τα οποια θα ειναι δυναμικα.Δηλαδη η κατασταση τους θα αλλαζει
//απο καποιους παραγοντες π.χ. πληκτρολογιο,ποντικι,χρονος,συναρτησεις κ.α)

          //Δημιουργία μεταβλητων.
      public Vector2 position; //ορίζει τη θεση του 2D αντικειμενου
      public float rotation = 0.0f; //ορίζει τη περιστροφη του 2D αντικειμενου
      public Texture2D spriteIndex; //μεταβλητη με τη διευθυνση που θα φορτώθουν τα
γραφικα.

      public string spriteName = "block";//Το μπλοκ ειναι μια εικονα που θα εμφανιζεται
στα....                              //...αντικειμενα που δεν τους εχουμε ορισει καποια αλλη
εικονα.

      public float speed = 0.0f; //Η ταχυτητα του αντικειμενου.
      public float Scale = 1.0f; //Οριζεται η κλιμακα.
      public Rectangle area; //
      //Οι μεταβλητες boolian δεχονται μονο δυο τιμες True–False μας χρησιμευει στο να
σταματαμε //τη συναρτηση update και draw οταν πεθανει το αντικειμενο.
      public bool alive = true; //Το αντικειμενο ειναι ζωντανο
      public bool solid = false; //


      public obj(Vector2 pos) //Συναρτηση obj.
      {
       position = pos; //θεση αντικειμενου.
      }

      public virtual void Update()
      {    //Οι συναρτησεις που εχουν την εντολη vitual, οπως στη παραπανω συναρτηση
//επηρεαζονται...(αλλαζουν τα δεδομενα τους) απο τις συναρτησεις με το ιδιο ονομα των
//αλλων αντικειμενων.
```

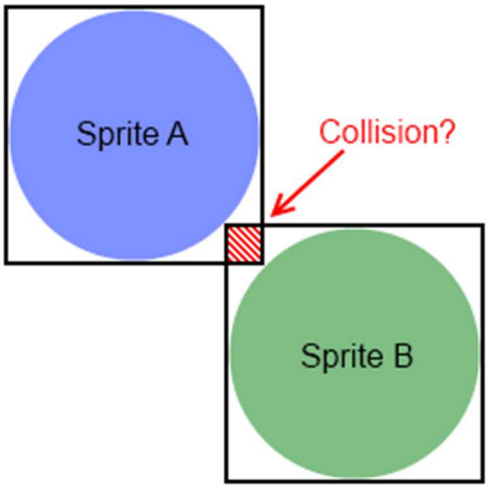
```
        if (!alive) return; //Αν δεν ειναι ζωντανο το αντικειμενο σταματα ειναι ειναι
δυναμικο. [6]


        pushTo(speed, rotation); ////Είναι ένας αλγόριθμος γεωμετρίας των πίξελ της οθόνης
ετσι ώστε να προσεγγίζεται η θέση του αντικειμένου στο παιχνίδι γνωρίζοντας τη κλίμακα
της ταχύτητας που θα κινείται και  θα περιστρέφεται.

//Εδω τοποθετειται ενας ελεγχος για το αν ειναι ζωντανο το αντικειμενο.

        if (heli.hp == 0)
        {       // Αν ο παικτης πεθανει, γυρναει το παιχνιδι στο Μενου, μηδενιζει το σκορ και
γεμιζει τη ζωη του για να ειναι ζωντανος για να επαναλαβει το γυρο, αμα ο χρηστης
πατησει την επιλογη start.
            BrainssssKiCKer.Gamestate = "Menu";
            heli. giveHp();
            heli.score = 0;
        }
        UpdateArea();//Ενημερωνει το τη θεση του αντικειμενου.
    }
    //φορτωνουμε το περιεχομενο του αντικειμενου.
    public virtual void LoadContent(ContentManager content)
    {
        //εδω δηλωνεται η διευθυνση που θα περιεχονται τα γραφικα που θα φορτωθουν
//στο παιχνιδι, στη περιπτωση αυτη ειναι ο φακελος "sprites\\"
        spriteIndex = content.Load<Texture2D>("sprites\\" + this.spriteName);
        area = new Rectangle(0 , 0 ,spriteIndex.Width , spriteIndex.Height );
        //Rectangle:Αποθηκεύει ένα σύνολο από τέσσερις ακέραιους που αντιπροσωπεύουν
ενα τετραγωνο με το μέγεθος του αντικειμενου μας.
    }
```



[10]

```
// Εδω ελεγχουμε τι θα εμφανιστει το περιβαλλον του παιχνιδιου
    public virtual void Draw(SpriteBatch spriteBatch)
    {
      if (!alive) return;

      //Δηλωνεται ποιο θα ειναι το κεντρο του αντικειμενου.
      Vector2 center = new Vector2(spriteIndex.Width/2, spriteIndex.Height/2);

      spriteBatch.Draw(spriteIndex, position, null, Color.White,
MathHelper.ToRadians(rotation), center, Scale,SpriteEffects.None, 0);   //Αυτη η εντολη
//χρειαζεται στην XNA για ξερει το προγραμμα σε ποιο σημειο, με ποιο μεγεθος ,τι
//αποχρωση,τι εφε θα εμφανισει τις εικονες του παιχνιδιου.
    }
```

[3] `// Η ιδεα του collision ειναι ενα τετραγωνο με το μεγεθος του παικτη (Rectangle) το //οποιο ακολουθει τον παικτη και εντοπιζει αν θα ακουμπισει μελλοντικα το τετραγωνο //απο καποιο αλλο αντικειμενο. Ο παρακατω κωδικας collision εμποδιζει τα αντικειμενα //απο τη λιστα να περασουν απο αλλα αντικειμενα της λιστας που ειναι σταθερα(solid) //π.χ.ο τοιχος.`




```
    public bool collision(Vector2 pos,obj obj) [6]
    {//Δημιουργειται μια καινουργια περιοχη που προσεγγιζει τη θεση του αντικειμενου.
      Rectangle newArea = new Rectangle(area.X, area.Y, area.Width, area.Height);
      newArea.X += (int)pos.X; //Η x-συντεταγμένη για την επάνω αριστερή γωνία του
//ορθογωνίου.
      newArea.Y += (int)pos.Y; //Η y-συντεταγμένη για την επάνω αριστερή γωνία του
//ορθογωνίου.

      foreach (obj MYobject in Items.objList)
      {//Γινεται ελεγχος για το αν καποι αντικειμενο απο τη λιστα ακουμπαει καποιο αλλο
//αντικειμενο της λιστα που ειναι στερεο (solid).
        if (MYobject.GetType() == obj.GetType() && MYobject.solid)
          if (MYobject.area.Intersects(newArea))
            return true;
      }
      return false;
```

```
}[6]
//Ο παρακατω κωδικας collision εξαφανιζει τις σφαιρες οταν πεφτουν πανω στον
    //εχθρο.
public obj collision(obj obj)
{
   foreach (obj MYobject in Items.objList)
   {
      if (MYobject.GetType() == obj.GetType() && MYobject.alive) //Το get type παίρνει
//τα δεδομενα για την καταταση των αντικειμενων που περιεχει η λιστα.
         if (MYobjecto.area.Intersects(area))
            return MYobject;
   }
    return new obj(Vector2.Zero);
}

public void UpdateArea()
{
   //φτιαχνουμε ενα τετραγωνο το οποιο ακολουθει τον παικτη για εντοπισουμε αν
//ακουμπαει καποια αλλο αντικειμενο.
   area.X = (int)position.X - (spriteIndex.Width/2  );
   area.Y = (int)position.Y - (spriteIndex.Height/2 );

}

public virtual void pushTo(float pix, float dir)
{    //Είναι ένας αλγόριθμος γεωμετρίας των πίξελ της οθόνης ετσι ώστε να
προσεγγίζεται η θέση του αντικειμένου στο παιχνίδι γνωρίζοντας τη κλίμακα της ταχύτητας
που θα κινείται και  θα περιστρέφεται.
   float newX = (float)Math.Cos(MathHelper.ToRadians(dir));
   float newY = (float)Math.Sin(MathHelper.ToRadians(dir));
   position.X += pix * (float)newX;
   position.Y += pix * (float)newY;
}

  }
}
```

ΑΝΤΙΚΕΙΜΕΝΟ:Heli.cs[3]

```
//Απαιτουμενες βιβλιοθηκες
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

namespace _2Dgame_project_A
{
   class heli : obj
   { //Σε αυτη τη ταξη υπαρχει ο κωδικας για τη κινηση,την περιστροφη,πυροβολισμος του
οπλου,


      //κατασταση πληκτρολογιου.
      KeyboardState keyboard;
      KeyboardState prevKeyboard;
      //κατασταση ποντικιου.
      MouseState mouse;
      MouseState prevMouse;

      public static int a = 0; //Αρχικοποιηση μεταβλητων. [6]
      public static int giros = 50;
      public static int d = 10;
      public static int e = 0;
      public static int hp = 100;
      static int hpMax = 100;
      public static int score = 0;

      // Ταχυτητα του παικτη.
      float spd;
      // Ταχυτητα των σφαιρων.
      float bSpd = 15;

      const int maxAmmo = 60; //μεγιστος αριθμος γεμιστηρα
      int ammo = 60; // αριθμος γεμιστηρα
      int rate =5; // Ρυθμος πυρος
      int firingTimer = 0;

      int reloadTimer = 0; //αρχικοποιηση μεταβλητης
      int reloadTime = 60; //χρονος για να οπλισει
      bool reloading = false; //κατασταση οπλισμα οπλου απενεργοποιημενη
```

```
        public static heli Heli;

        public heli(Vector2 pos)
            :base(pos) //βρισκει τη θεση απο τη βαση δεδομενων.
        {
            position = pos;
            spd = 5; //Η ταχυτητα κινησης του παικτη.
            spriteName = "player";
            Heli = this;

        }

        public static void giveHp()[3]
        {
            hp = hpMax; //Δινει μεγιστη ζωη
        }


        public static void RemoveHp()
        {
            hp -- ; //Μειώνει τη ζωη του παικτη κατα 1
        }

        public static Boolean hasHp()
        {
            return hp > 0; //Ελεγχει αν εχει ζωη ο παικτης.
        }

        public static void increaseScore()
        {
            score+=5; //Αυξανει το σκορ κατα 5
        }

        //Η συναρτηση που ειναι override μπορει να ενημερωνει τις τιμες των μεταβλητων σε
    //συναρτησεις με το ιδιο ονομα που ειναι virtual(πχ στη ταξη obj ) που βρισκονται σε αλλη
    //ταξη .
        public override void Update()
        {[7]

            Heli = this;
            //κωδικας για τον ελεγχο του οπλου απο το πληκτρολογιο και το ποντικι.
            keyboard = Keyboard.GetState();//εκχωρει την κατασταση του πληκρολογιου στη
    //μεταβλητη.
            mouse = Mouse.GetState();//εκχωρει την κατασταση του ποντικιου.

            //Κινηση του χαρακτηρα απο τα βελακια
            if (keyboard.IsKeyDown(Keys.Up) && !collision(new Vector2(0, -spd), new Wall(new
    Vector2(0, 0))))
            {//Εαν το πληκτρο 'ΠΑΝΩ' εχει πατηθει ΚΑΙ εαν ο παικτης ΔΕΝ ακουμπαει τον τοιχο
    //τοτε μειωσε τη θεση του αντικειμενου κατα spd.Το ιδιο γινεται για τις παρακατω εντολες
    //των παρακατω πληκτρων.
```

```
            position.Y -= spd; [7]
        }
        if (keyboard.IsKeyDown(Keys.Left) && !collision(new Vector2(-spd, 0), new Wall(new
Vector2(0, 0))))
        {
            position.X -= spd;
        }
        if (keyboard.IsKeyDown(Keys.Down) && !collision(new Vector2(0, spd), new Wall(new
Vector2(0, 0))))
        {
            position.Y += spd;
        }
        if (keyboard.IsKeyDown(Keys.Right) && !collision(new Vector2(spd, 0), new Wall(new
Vector2(0, 0))))
        {
            position.X += spd;
        }

        //Κινηση του οπλου απο τα(W.A.S.D)
        if (keyboard.IsKeyDown(Keys.W) && !collision(new Vector2(0, -spd), new Wall(new
Vector2(0,
        {
            position.Y -= spd;
        }
        if (keyboard.IsKeyDown(Keys.A) && !collision(new Vector2(-spd, 0), new Wall(new
Vector2(0,
        {
            position.X -= spd;
        }
        if (keyboard.IsKeyDown(Keys.S) && !collision(new Vector2(0, spd), new Wall(new
Vector2(0,
        {
            position.Y += spd;
        }

        if (keyboard.IsKeyDown(Keys.D) && !collision(new Vector2(spd, 0), new Wall(new
Vector2(0,
        {
            position.X += spd;
        }
        firingTimer++;
        //οταν γεμιζε το οπλο δεν πυροβολαει(!reloading)!!!
        if (mouse.LeftButton == ButtonState.Pressed && !reloading)
        {
            CheckShooting();//Οταν ενεργοποιειται η συναρτηση το οπλο πυροβολαει.

        }
        if (keyboard.IsKeyDown(Keys.Space) && !reloading)
        {
            CheckShooting();//Οταν ενεργοποιειται η συναρτηση το οπλο πυροβολαει.
        }
```

```
        //Εντολη πληκτρολογιου για να γεμισει το οπλο σφαιρες.       [7]
        if (keyboard.IsKeyDown(Keys.R)|| keyboard.IsKeyDown(Keys.NumPad0))
        {
           reloading = true;
        }
        //Εντολη πληκτρολογιου για να γεμισει το οπλο σφαιρες.
        if (keyboard.IsKeyDown(Keys.RightControl)|| keyboard.IsKeyDown(Keys.LeftControl))
        {
           reloading = true;
        }
        CheckReload();//Η συναρτηση ελεγχει τον γεμισμα του οπλου.

        //Εαν ο γεμιστηρας αδειασει γεμισει το οπλο σφαιρες.       [6]
        if (ammo == 0)
        {
           reloading = true;
        }
        CheckReload();//Η συναρτηση ελεγχει τον γεμισμα του οπλου.

        //Συνδεει τη περιστροφη του οπλου με τη θεση του ποντικιου ετσι ωστε το οπλο να
//σημαδευει τον κερσορα του ποντικιού.
        rotation = point_direction(position.X, position.Y, mouse.X, mouse.Y);

        prevKeyboard = keyboard; //εκχωρει την προηγουμενη κατασταση του
//πληκρολογιου.
        prevMouse = mouse; //εκχωρει την προηγουμενη κατασταση του ποντικιου.
        base.Update();//Ενημερωση της βασης δεδομενων.

        //Αλλαζει το Gamestate στο Menu για να κανουμε ΠΑΥΣΗ.
        if (keyboard.IsKeyDown(Keys.P) ) //Παυση με το πληκτρο 'P'
        {
           BrainssssKiCKer.Gamestate = "Menu";

        }

        if (keyboard.IsKeyDown(Keys.Back)) //Παυση με το πληκτρο 'πισω'
        {
           BrainssssKiCKer.Gamestate = "Menu";

        }
        if (hp <= 0) //εαν η ζωη το παικτη γινει μηδεν τοτε
        {
           hp = 0; //εαν ζωη ειναι μηδεν σε περιπτωση που παρει αρνητικη τιμη.
           alive = false; //το αντικειμενο παυει να ειναι δυναμικο.
        }

        //Εδω δημιουργουνται τα διαφορετικα σκορ τις καθε πιστας.
//Εδω δημιουργω τα διαφορετικα σκορ τις καθε πιστας.

        for (int z = 0; z < 11; z++)//Επαναληψη 10 γυρων.
        {
```

```
            if (z == a)//ελεγχει σε ποιο γυρο ειναι το παιχνιδι. [3]
            {
               if (score == giros)//εαν το σκορ γινει ισο με το γυρο εκτελουνται τα παρακατω.
               {
                  //algorithm for time.
                  BrainssssKiCKer.Gamestate = "Menu";//αλλαζει τη καταστση του παιχνιδιου
                                                      //σε καταστση μενου.

                  d = d - 1;//ποσοι γυροι εχουν μεινει.
                  giros = giros + 200;//algorithm for score.
                  score = 0;//μηδενιζεται το σκορ.
                  a = a + 1;//αυξανετε ο γυρος κατα 1.
                  heli.giveHp();//Μολις τελιωσει ο γυρος γεμιζει ζωη τον παικτη.
                  foreach (obj MyZombies in Items.objList)
                              //Για καθε αντικειμενο της λιστας κανε.
                  {
                     if (MyZombies.GetType() == typeof(Enemy))//οποιο απο τα αντικειμενα
                                                      //ειναι τυπος εχθρος κανε.
                     {
                        MyZombies.alive = false;
                              //Μολις τελειωσει ο γυρος αδειαζει η λιστα με τα ζομπι.
                     }
                  }
               }
            }
         }
         if (a == 11)//Αν ο γυρος παει 11 γυρνα στο μενου  το παιχνιδι τελιωσε.
         {
            BrainssssKiCKer.Gamestate = "Menu";//αλλαζει τη καταστση του παιχνιδιου σε
                                                //καταστση μενου.
         }
      }
      //Εμφανιζει στη οθονη καποιες πληροφορες σε σχεση με την καταστση του
//παιχνιδιου οπως ammo, health, score....

      public override void Draw(SpriteBatch spriteBatch)
      {

         if (BrainssssKiCKer.Gamestate == "Game")
         {[4]
            //Ο παρακατω κωδικας εμφανιζει στην οθονη την καταστστη των Ammo kai
//Reloading score ,pause ,health kai round.
            spriteBatch.DrawString(BrainssssKiCKer.font, "Health :" + hp, new Vector2(5, -3 +
BrainssssKiCKer.font.LineSpacing), Color.GreenYellow);

            spriteBatch.DrawString(BrainssssKiCKer.font, "Ammo :" + ammo, new Vector2(5, -
30 + BrainssssKiCKer.font.LineSpacing), Color.GreenYellow);

            spriteBatch.DrawString(BrainssssKiCKer.font, "Score :" + score + "/" + giros, new
Vector2(5, 25 + BrainssssKiCKer.font.LineSpacing), Color.GreenYellow);
```

```
[4]         spriteBatch.DrawString(BrainssssKiCKer.font, "Press 'P' or 'Back' for Pause Game",
new Vector2(500, -15 + BrainssssKiCKer.font.LineSpacing), Color.GreenYellow);

            spriteBatch.DrawString(BrainssssKiCKer.font, "Round " + a, new Vector2(150, -30 +
BrainssssKiCKer.font.LineSpacing), Color.GreenYellow);

            if (reloading)
            {
               spriteBatch.DrawString(BrainssssKiCKer.font, "Reloading :", new Vector2(10, 70 +
BrainssssKiCKer.font.LineSpacing), Color.Red);
            }
            base.Draw(spriteBatch);
        }

    }[6]

    //Η συναρτηση ελεγχει τον γεμισμα του οπλου.
    private void CheckReload()
    {
       if (reloading)
          reloadTimer++;

       if (reloadTimer > reloadTime)
       {
          ammo = maxAmmo;
          reloadTimer = 0;
          reloading = false;
       }
    }
    //Η συναρτηση ελεγχει ποτε και ποσο πυροβολαει το οπλο.
    private void CheckShooting()
    {//ελεγχει ποτε θα αδιασει το οπλο "ammo >0".
       if (firingTimer > rate && ammo >0)
       {
          firingTimer = 0;
          Shoot();

       }
    }

    private void Shoot()
    {
       ammo--;//Οσο πυροβολαει το οπλο μειωνονται οι σφαιρες.

       foreach (obj MYobject in Items.objList) //Για καθε αντικειμενο της λιστας κανε...
       { // με τον παρακατω ελεγχο επιτρεπουμε στο οπλο να πυροβολαει οταν εχει
         //σφαιρες και ειναι ζωντανο.
          if(MYobject.GetType() == typeof(Bullets) && ! MYobject.alive)
          {//Εαν καποιο My object της λιστας ειναι σφαιρες και δεν ειναι ηδη μεσα στο
          //  παιχνιδι κανε τα παρακατω .
             MYobject.position = position; //Η σφαιρα ξεκιναει απο τη θεση του οπλου.
```

```
                MYobject.UpdateArea();//Ενημερωνει την θεση της σφαιρας.
                MYobject.rotation = rotation; //Η σφαιρα παιρνει περιστροφη  του οπλου
                //δηλωνουμε τη ταχυτητα στης σφαιρας.
                MYobject.speed = bSpd; //Η σφαιρα παιρνει τη ταχυτητα bspd.
                MYobject.alive = true; //Η σφαιρα ειναι ζωντανη.
                break;
            }
        }
    }

//Ο παρακατω αλγοριθμος βρισκει τη γωνια που σημαδευεις με το ποντικι. [6]
    private float point_direction(float x, float y, float x2, float y2)
    {
        float diffx = x - x2;
        float diffy = y - y2;
        float adj = diffx;
        float opp = diffy;
        float tan = opp / adj;
        float res = MathHelper.ToDegrees((float)Math.Atan2(opp,adj));
        res = (res - 180) % 360;
        if (res < 0) { res += 360; }
        return res;

    }

  }
}
```

ΑΝΤΙΚΕΙΜΕΝΟ: Items.cs[3]

```
//Απαιτουμενες βιβλιοθηκες
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

namespace _2Dgame_project_A
{
   class Items
   {
      public static List<obj> objList = new List<obj>();//Δημιουργια μιας λιστας με ολα τα
                                                         //αντικειμενα.

     //Η συναρτηση Initialize()  προετοιμάζει το περιβάλλον του παιχνιδιού π.χ. ορίζει την
//αρχική κατάσταση των αντικειμένων

      public static void Initilize()
      {
            for (int i = 0; i < 80; i++)
            {  //στη επαναληψη αυτη οριζουμε 80 'ζωντανες' σφαιρες.
               obj MYobject = new Bullets(new Vector2(0,0));
               MYobject.alive = false; //Τα αντικειμενα ειναι στη καταταση ζωντανα
               objList.Add(MYobject);
            }
```

[6]

```
            //Μεγιστος αριθμος ζωντανων εχθρων που θα εμφανιζονται στο παιχνιδι ειναι 60.
.
            for (int j = 0; j < 60; j++)
            {
               Enemy MyZombies = new Enemy(new Vector2(1340, 90));
               MyZombies.alive = false;
               objList.Add(MyZombies);
             }
             //εδω βαζουμε τον παικτη και τον στοχο στο παιχνιδι.

            objList.Add(new heli(new Vector2(350, 450)));

            objList.Add(new Cursor(new Vector2(0, 0)));

            //εμφανιζει τα σημειου που θα βγαινει ο εχθρος.
```

```
objList.Add(new ZombieSpawner(new Vector2(1550, 600)));

objList.Add(new ZombieSpawner(new Vector2(1850, 300)));

objList.Add(new ZombieSpawner(new Vector2(1650, 100)));

objList.Add(new ZombieSpawner(new Vector2(1750, 900)));
//Εδω φτιαχνεται ενα πλαισιο απο τοιχο στο παιχνιδι.

//Πλαισιο με γυρους,σκορ,ζωη,σφαιρες.
objList.Add(new Wall(new Vector2(230, 50)));
objList.Add(new Wall(new Vector2(200, 50)));
objList.Add(new Wall(new Vector2(115, 50)));
objList.Add(new Wall(new Vector2(50, 50)));
//αριστερα τοιχος
objList.Add(new Wall(new Vector2(-40, 150)));
objList.Add(new Wall(new Vector2(-40, 250)));
objList.Add(new Wall(new Vector2(-40, 350)));
objList.Add(new Wall(new Vector2(-40, 450)));
objList.Add(new Wall(new Vector2(-40, 550)));
objList.Add(new Wall(new Vector2(-40, 650)));
objList.Add(new Wall(new Vector2(-40, 750)));
objList.Add(new Wall(new Vector2(-40, 850)));
objList.Add(new Wall(new Vector2(-40, 950)));

//δεξια τοιχος
objList.Add(new Wall(new Vector2(1570, 50)));
objList.Add(new Wall(new Vector2(1570, 150)));
objList.Add(new Wall(new Vector2(1570, 250)));
objList.Add(new Wall(new Vector2(1570, 350)));
objList.Add(new Wall(new Vector2(1570, 450)));
objList.Add(new Wall(new Vector2(1570, 550)));
objList.Add(new Wall(new Vector2(1570, 650)));
objList.Add(new Wall(new Vector2(1570, 750)));
objList.Add(new Wall(new Vector2(1570, 850)));
objList.Add(new Wall(new Vector2(1570, 950)));

objList.Add(new Wall(new Vector2(1670, 50)));
objList.Add(new Wall(new Vector2(1670, 150)));
objList.Add(new Wall(new Vector2(1670, 250)));
objList.Add(new Wall(new Vector2(1670, 350)));
objList.Add(new Wall(new Vector2(1670, 450)));
objList.Add(new Wall(new Vector2(1670, 550)));
objList.Add(new Wall(new Vector2(1670, 650)));
objList.Add(new Wall(new Vector2(1670, 750)));
objList.Add(new Wall(new Vector2(1670, 850)));
objList.Add(new Wall(new Vector2(1670, 950)));
objList.Add(new Wall(new Vector2(1670, 1050)));
```

```
            //πανω τοιχος

            objList.Add(new Wall(new Vector2(1400, -40)));
            objList.Add(new Wall(new Vector2(1300, -40)));
            objList.Add(new Wall(new Vector2(1200, -40)));
            objList.Add(new Wall(new Vector2(1100, -40)));
            objList.Add(new Wall(new Vector2(1000, -40)));
            objList.Add(new Wall(new Vector2(900, -40)));
            objList.Add(new Wall(new Vector2(800, -40)));
            objList.Add(new Wall(new Vector2(700, -40)));
            objList.Add(new Wall(new Vector2(600, -40)));
            objList.Add(new Wall(new Vector2(500, -40)));
            objList.Add(new Wall(new Vector2(400, -40)));
            objList.Add(new Wall(new Vector2(300, -40)));
            objList.Add(new Wall(new Vector2(200, -40)));
            objList.Add(new Wall(new Vector2(100, -40)));
            objList.Add(new Wall(new Vector2(0, -40)));

            //κατω τοιχος
            objList.Add(new Wall(new Vector2(1600, 1040)));
            objList.Add(new Wall(new Vector2(1500, 1040)));
            objList.Add(new Wall(new Vector2(1400, 1040)));
            objList.Add(new Wall(new Vector2(1300, 1040)));
            objList.Add(new Wall(new Vector2(1200, 1040)));
            objList.Add(new Wall(new Vector2(1100, 1040)));
            objList.Add(new Wall(new Vector2(1000, 1040)));
            objList.Add(new Wall(new Vector2(900, 1040)));
            objList.Add(new Wall(new Vector2(800, 1040)));
            objList.Add(new Wall(new Vector2(700, 1040)));
            objList.Add(new Wall(new Vector2(600, 1040)));
            objList.Add(new Wall(new Vector2(500, 1040)));
            objList.Add(new Wall(new Vector2(400, 1040)));
            objList.Add(new Wall(new Vector2(300, 1040)));
            objList.Add(new Wall(new Vector2(200, 1040)));
            objList.Add(new Wall(new Vector2(100, 1040)));
            objList.Add(new Wall(new Vector2(0, 1040)));
        }
    }
}
```

ANTIKEIMENO: Enemy.cs[3]

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

namespace _2Dgame_project_A
{
   class Enemy : obj[6]
   {//Αρχικοποιηση καποιων μεταβλητων που θα μας χρησιμευσουν παρακατω.
      int health = 100;
      float spd = 1; //ταχυτητα του ζομπι.

      int damageTimer = 300; //Οταν η δίπλα μεταβλητη ειναι μηδεν  το ζομπι μπορει να //
                                    // τραυματισει τον παικτη , το 300 ειναι ενα ειδος
                        //καθυστέρησης για καθε χτυπημα του ζομπι.

      public Enemy(Vector2 pos)
         : base(pos)
      {
         position = pos; //η θεση του ζομπι
         spriteName = "zombie";//το ονομα του αρχειου του ζομπι.
         health = 50; //η ζωη του ζομπι
         speed = spd; //ταχυτητα του ζομπι.
         if (!alive) return;

      }
      public void damagePlayer()
      {
         if (damageTimer == 0)//Οταν ειναι μηδεν  το ζομπι μπορει να τραυματισει τον παικτη
         {
            damageTimer = 300;
            heli.RemoveHp();//Μειώνει τη ζωη του παικτη κατα 1
         }
      }
```

```
public override void Update()[6]
{

    //Ο εχθρος ερχεται προς το μερος του παικτη , οταν του τελιωσει η ζωη
    //εξαφανιζεται.
   if (health > 0)
   {
      rotation = point_direction(position.X, position.Y, heli.Heli.position.X ,
heli.Heli.position.Y); //Το ζομπι πηγαινει προς την θεση του παικτη.
   }

   if (damageTimer > 0)
      damageTimer--; //Μειωνει κατα 1 το χρονο.

   //Οταν ο εχθρος ''ακουμπησει'' τον παικτη, ο παικτης χανει ζωη

   obj MyZombies = collision(new heli(new Vector2(heli.Heli.position.X ,
heli.Heli.position.Y )));

   if (MyZombies.GetType() == typeof(heli)) //Εαν το αντικειμενο ειναι ο παικτης τοτε
   {
      damagePlayer();//καλει τη συναρτηση που μειωνε ιτη ζωη του παικτη.
   }

  // Οταν πεθανει ενας εχθρος παιρνεις 5 ποντους.
   if (health == 0 )
   {
      heli.increaseScore();//Η συναρτηση αυτη καλειται απο το αντικειμενο heli και
αυξανει το...                    //...σκορ κατα 5 μοναδες.

      alive = false; //Το αντικειμενο δεν ειναι δυναμικο πια γιατι δεν ειναι ζωντανο.

      health = 50; //Η ζωη του εχθρου zombie.
   }

   base.Update();// Ενημερωνετε η βαση δεδομενων.

}
```

```
        //Ο παρακατω αλγοριθμος βρισκει τη γωνια που σημαδευεις με το ποντικι.
        private float point_direction(float x, float y, float x2, float y2) [6]
        {
            float diffx = x - x2;
            float diffy = y - y2;
            float adj = diffx;
            float opp = diffy;
            float tan = opp / adj;
            float res = MathHelper.ToDegrees((float)Math.Atan2(opp, adj));
            res = (res - 180) % 360;
            if (res < 0) { res += 360; }
            return res;
        }

        public void Damage(int dmg)
        {
            health -= dmg; Μειωνει τη ζωη των ζομπι κατα 1

        }

    }
}
```

ANTIKEIMENO: ZombieSpawner.cs[3]

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

namespace _2Dgame_project_A
{
   class ZombieSpawner : obj[6]
   {//Αρχικοποιησεις μεταβλητων
      private int spawnTimer = 0;
      public static int spawnTime = 300;

      public ZombieSpawner(Vector2 pos)
         : base(pos)
      {
         position = pos; //Η θεση του αντικειμενου
      }

      //Απο τη λιστα με τα αντικειμενα που δημιουργησα παιρνει τον εχθρο.

      public override void Update()
      {
         IncrementTimers();//Αυξανει κατα ενα το χρονο

         if (spawnTimer > spawnTime) //καθε φορα που το spawnTimer γινεται 300 βγαινει
                                     //ενα ζομπι.
         {
            spawnTimer = 0;

            foreach (obj MYobject in Items.objList)
            {
               if (MYobject.GetType() == typeof(Enemy) && ! MYobject.alive)
               {//εαν βρει ζομπι παει στη κατω εντολη
                  MYobject.alive = true; //Ζωντανευει ενα ζομπι
                  MYobject.position = position; //Ενα ζομπι εμφανιζεται στη θεση του zombie
spawner.
                  break; //Διακοπτουμε την εντολη for each με το break για να βγαζει καθε
                         //φορα που το spawnTime γινεται 300 ενα ζομπι.
               }
            }
         }
      }
```

```
    private void IncrementTimers()[3]
    {
        if (heli.a == 1 || heli.a == 2 || heli.a == 3) //Η ταχυτητα εμφανισης ζομπι στου γυρους
1,2,3
        {
            spawnTimer = spawnTimer + 1;
        }
        if (heli.a == 4 || heli.a == 5 || heli.a == 6 || heli.a == 7)
        {//Η ταχυτητα εμφανισης ζομπι στου γυρους 4,5,6
            spawnTimer = spawnTimer + 2;
        }
        if (heli.a == 8 || heli.a == 9 || heli.a == 10) //Η ταχυτητα εμφανισης ζομπι στου
γυρους 8,9,10
        {
            spawnTimer = spawnTimer + 3;
        }
    }
  }
}
```

ΑΝΤΙΚΕΙΜΕΝΟ: Bullets.cs[3]

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

namespace _2Dgame_project_A
{
   class Bullets : obj[6]
   {
      public Bullets(Vector2 pos)
         : base(pos) //βρισκει τη θεση απο τη βαση δεδομενων του obj.
      {
         position = pos;
         spriteName = "bullet";
      }

       public override void Update()
       {
          if (!alive) return; //Αν δεν ειναι ζωντανη δεν κανει τιποτα.

          //εξαφανιζεται η σφαιρα οταν πεφτει σε τοιχο.
          if (collision(Vector2.Zero, new Wall(new Vector2(0, 0))))
          {
             alive = false; //η σφαιρα δεν ειναι ζωντανη.
          }

          //Δημιουργει damage στον εχθρο οταν ερχεται σε επαφη με τη σφαιρα.

          obj MYobject = collision(new Enemy(new Vector2(0, 0)));

          if (MYobject.GetType() == typeof(Enemy)) //Αν το αντικειμενο ειναι εχθρος κανε...
          {
             alive = false;
             Enemy MyZombies = (Enemy) MYobject; //Μετατροπη obj σε Enemy.
             MyZombies.Damage(10); //Μειωνει τη ζωη των ζομπι κατα 1 απο τα 10
          }
          base.Update();//Ενημερωση της βασης δεδομενων.
       }

   }
}
```

ANTIKEIMENO: Wall.cs[3]

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

namespace _2Dgame_project_A
{
   //Δημιουργω τοιχο στο παιχνιδι.
   class Wall : obj
   {

      public Wall(Vector2 pos) [6]
         : base(pos) //βρισκει τη θεση απο τη βαση δεδομενων του obj
      {
         solid = true; //ειναι στερεο.
         position = pos; //μεταβλητη για τη θεση του τοιχου .
         spriteName = "wall";//το ονομα του αρχειου  του τοιχου .
      }
   }
}
```

ΑΝΤΙΚΕΙΜΕΝΟ: Cursor.cs[3]

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

namespace _2Dgame_project_A
{
   class Cursor : obj[6]
   {  //Σε αυτη τη ταξη υπαρχει ο κωδικας του κερσορα "στοχου του οπλου".
      MouseState mouse; //εκχωρει την κατασταση του ποντικιου.
      //Αυτη η ταξη επιστρεφει τη θεση του κερσορα στο παιχνιδι και προς αυτη τη θεση θα
...         //...κατευθυνεται η σφαιρα που ξεκιναει απο το παικτης-οπλο.
      public Cursor(Vector2 pos)
         : base(pos) //βρισκει τη θεση απο τη βαση δεδομενων.
      {
         position = pos; //Η θεση του αντικειμενου.
         spriteName = "cursor";//Το ονομα του αρχειου της εικονας του κερσορα.
      }

      public override void Update()
      {
         mouse = Mouse.GetState();//Η κατασταση που βρισκεται το ποντικι.

         position = new Vector2(mouse.X,mouse.Y); // τη θεση του κερσορα στο παιχνιδι.

         base.Update();//Ενημερωση της βασης δεδομενων.
      }


   }
```

ANTIKEIMENO: Menu.cs[3]

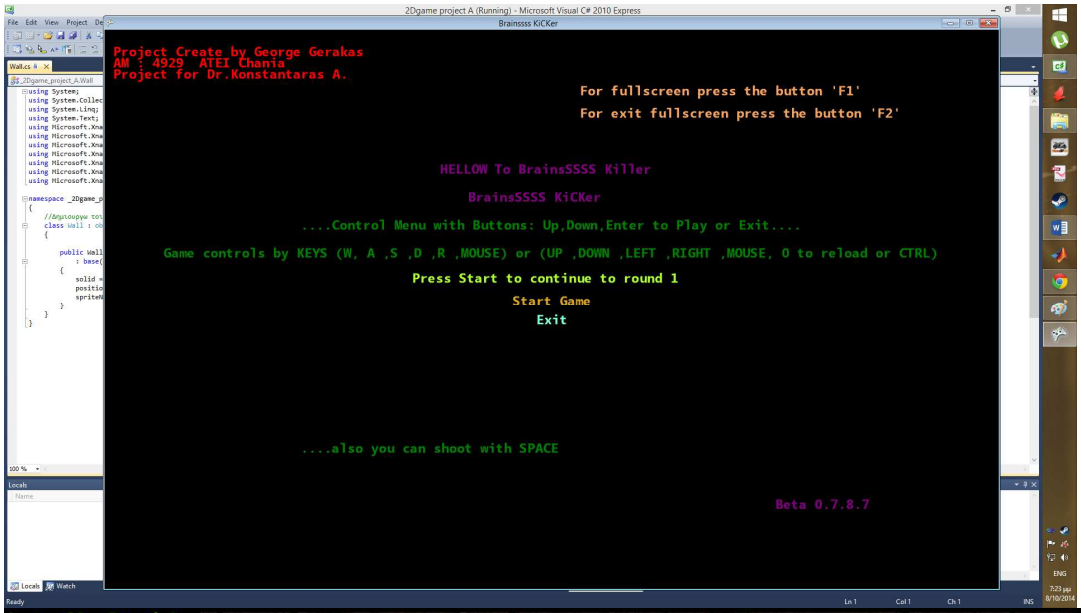
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

namespace _2Dgame_project_A
{
   class Menu[6]
   {
      public static int c = 0;

      KeyboardState keyboard; //κατασταση πληκτρολογιου.
      KeyboardState prevKeyboard; //προηγουμενη κατασταση πληκτρολογιου.

      SpriteFont spriteFont; // αρχικοποιηση της μεταβλητης ΛΕΞΕΩΝ.

      List<string> buttonList = new List<string>();// Δημιουργια λιστας κουμπιων μενου.

      int selected = 0; // αρχικοποιηση της μεταβλητης

      public Menu()
      {// μενου κουμπια μπορουν να αλλαζουν την κατασταση του παιχνιδιου.
         buttonList.Add("Start Game");// μενου κουμπι 'startgame'.
         buttonList.Add("Exit");// μενου κουμπι 'exit'.
      }

      public void LoadContent(ContentManager Content)
      {
         spriteFont = Content.Load<SpriteFont>(" Grammatoseira ");//φορτωνει το αρχειο
      }
```

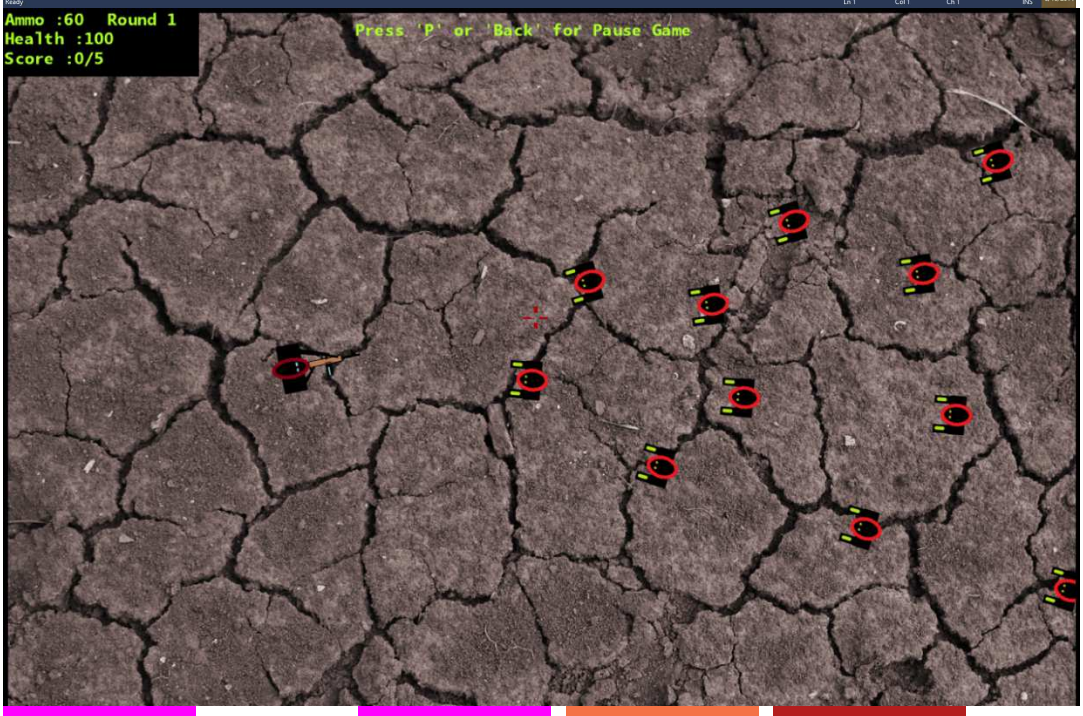
```
        public void Update(GameTime gameTime) [7]
        {
            c = heli.α - 1; //Θετει στο c τον αριθμο του προηγουμενου γυρου.
            keyboard = Keyboard.GetState();

             //Ελεγχος του μενου απο το πληκτρολογιο.
            if (CheckKeyboard(Keys.Up))
            {//Εαν πατηθει το πληκτρο πανω κανε
                if (selected > 0) selected--;
//Εαν το selected ειναι μεγαλυτερο απο το 0 μειωσε το κατα 1.
            }
            if (CheckKeyboard(Keys.Down))
            {//Εαν πατηθει το πληκτρο κατω κανε
                if (selected < buttonList.Count - 1) selected++;
//Εαν το selected ειναι μικροτερο απο το τελευταιο κουμπι αυξησε το κατα 1.

            }
            if (CheckKeyboard(Keys.Enter)) //Εαν πατηθει το πληκτρο enter κανε.
            {
                switch (selected) //επελεξε τον αριθμο της μεταβλητης selected.
                {
                    case 0: //Εαν το selected ειναι 0 ξεκινα το παιχνιδι.
                        BrainssssKiCKer.Gamestate ="Game";
                        break;

                    case 1: //Εαν το selected ειναι 1 πηγαινε στο μενου.
                        BrainssssKiCKer.Gamestate = "Exit";
                        break;
                }
            }

            if (CheckKeyboard(Keys.Space)) //Εαν πατηθει το πληκτρο space κανε.
            {
                switch (selected) //επελεξε τον αριθμο της μεταβλητης selected.
                {
                    case 0: //Εαν το selected ειναι 0 ξεκινα το παιχνιδι.
                        BrainssssKiCKer.Gamestate = "Game";
                        break;

                    case 1: //Εαν το selected ειναι 1 πηγαινε στο μενου.
                        BrainssssKiCKer.Gamestate = "Exit";
                        break;
                }
            }
        }

        public bool CheckKeyboard(Keys key)
        {//Ελεγχος για το εαν  εχει πατηθει καποιο πληκτρο απο το πληκτρολογιο του χρηστη.
            return (keyboard.IsKeyDown(key) && !prevKeyboard.IsKeyDown(key));
        }
```

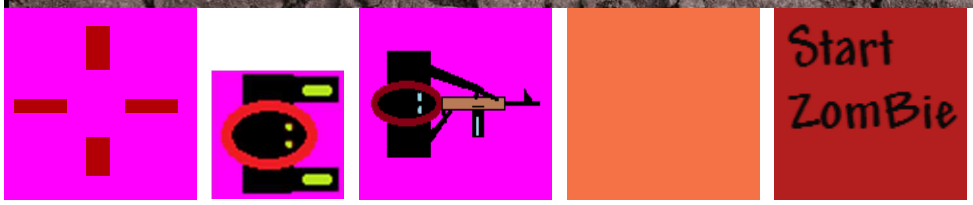
```
 public void Draw(SpriteBatch spriteBatch)
      {
         Color color; //Δηλωση της μεταβλητης
         int linePadding = 10; //κενο γραμμης μεταξυ των κουμπιων "Exit" και ''Game'' .

//Στην XNA για να εμφανιστουν καποια γραφικα στο παιχνιδι με τη συναρτηση  draw ειναι
//απαραιτητο ο κωδικας για την φωτογραφια να μπει αναμεσα στις εντολες
//spriteBatch.Begin() και spriteBatch.End().[5]

         spriteBatch.Begin();


         for (int i = 0; i < buttonList.Count; i++)//Για οσα κουμπια εχει η λίστα κανε
         {
//Το χρωμα για οταν το κουμπι ειναι επιλεγμενο και το χρωμα για οταν το κουμπι δεν ειναι
//ειναι επιλεγμενο
            color = (i == selected) ? Color.Goldenrod : Color.Aquamarine;
//Ο κατω κωδικας εμφανιζει τα κουμπια στο μενου.
            spriteBatch.DrawString(spriteFont, buttonList[i], new Vector2
             ((BrainssssKiCKer.screen.Width / 2) - (spriteFont.MeasureString(buttonList[i]).X / 2),
          (BrainssssKiCKer.screen.Height / 2) - (spriteFont.LineSpacing * buttonList.Count / 2) +
           ((spriteFont.LineSpacing + linePadding) * i)), color);
         }
//Τα παρακατω μηνυματα εμφανιζονται μονο στο μενου
         if (BrainssssKiCKer.Gamestate == "Menu")
         {

      //Ο παρακατω κωδικας εμφανιζει στην οθονη την καταταση καποιων μεταβλητων
//και δινει διαφορες οδηγιες στον χρηστη για το πως να χειριστει το παιχνιδι. [4]

            spriteBatch.DrawString(BrainssssKiCKer.font, "Project Create by Gerakas George
Hlias", new Vector2(15, -10 + BrainssssKiCKer.font.LineSpacing), Color.Red);

            spriteBatch.DrawString(BrainssssKiCKer.font, "AM : 4929  ATEI Chania", new
Vector2(15, 10 + BrainssssKiCKer.font.LineSpacing), Color.Red);

            spriteBatch.DrawString(BrainssssKiCKer.font, "Project for Dr.Konstantaras A.", new
Vector2(15, 30 + BrainssssKiCKer.font.LineSpacing), Color.Red);

            spriteBatch.DrawString(BrainssssKiCKer.font, "HELLOW To BrainsSSSS KiCKer ", new
Vector2(600, 200 + BrainssssKiCKer.font.LineSpacing), Color.Purple);

            spriteBatch.DrawString(BrainssssKiCKer.font, "Beta 0.7.8.7", new Vector2(1200, 800
+ BrainssssKiCKer.font.LineSpacing), Color.Purple);

            spriteBatch.DrawString(BrainssssKiCKer.font, "....Control Menu with Buttons:
Up,Down,Enter to Play or Exit....", new Vector2(350, 300 + BrainssssKiCKer.font.LineSpacing),
Color.Green);
```

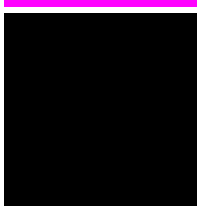
```
[4]        spriteBatch.DrawString(BrainssssKiCKer.font, "Game controls by KEYS (W, A ,S ,D ,R
,MOUSE) or (UP ,DOWN ,LEFT ,RIGHT ,MOUSE, 0 to reload or CTRL)", new Vector2(105, 350
+ BrainssssKiCKer.font.LineSpacing), Color.Green);

            spriteBatch.DrawString(BrainssssKiCKer.font, "....also you can shoot with SPACE",
new Vector2(350, 700 + BrainssssKiCKer.font.LineSpacing), Color.Green);

            spriteBatch.DrawString(BrainssssKiCKer.font, "For fullscreen press the button 'F1' ",
new Vector2(850, 60 + BrainssssKiCKer.font.LineSpacing), Color.SandyBrown);

            spriteBatch.DrawString(BrainssssKiCKer.font, "For exit fullscreen press the button
'F2' ", new Vector2(850, 100 + BrainssssKiCKer.font.LineSpacing), Color.SandyBrown);

           if (heli.a != 11) //Το μηνυμα εμφανιζεται μεχρι να τερματιστει το παιχνιδι.
            {
               spriteBatch.DrawString(BrainssssKiCKer.font, "Press Start to continue to round " +
heli.a , new Vector2(550, 395 + BrainssssKiCKer.font.LineSpacing), Color.GreenYellow);
            }


            if (heli.a != 11 && heli.a != 1)
            {//Το μηνυμα εμφανιζεται απο το τελος του 1 γυρου μεχρι να τελιωσει παιχνιδι.
               spriteBatch.DrawString(BrainssssKiCKer.font, "Congratulation you finish the
round " + c + "!!!! ||" + heli.d + " Rounds Remaining!!!!", new Vector2(350, 600 +
BrainssssKiCKer.font.LineSpacing), Color.GreenYellow);
            }

            if (heli.a == 11) //Το μηνυμα εμφανιζεται μολις τερματιστει το παιχνιδι.
            {
               spriteBatch.DrawString(BrainssssKiCKer.font, "Congratulation you win :)!!! ", new
Vector2(650, 550 + BrainssssKiCKer.font.LineSpacing), Color.GreenYellow);
               spriteBatch.DrawString(BrainssssKiCKer.font, "No more rounds!!!You can only
exit the game!!! ", new Vector2(550, 650 + BrainssssKiCKer.font.LineSpacing),
Color.GreenYellow);
            }
         }
         spriteBatch.End();
      }
   }
}
```

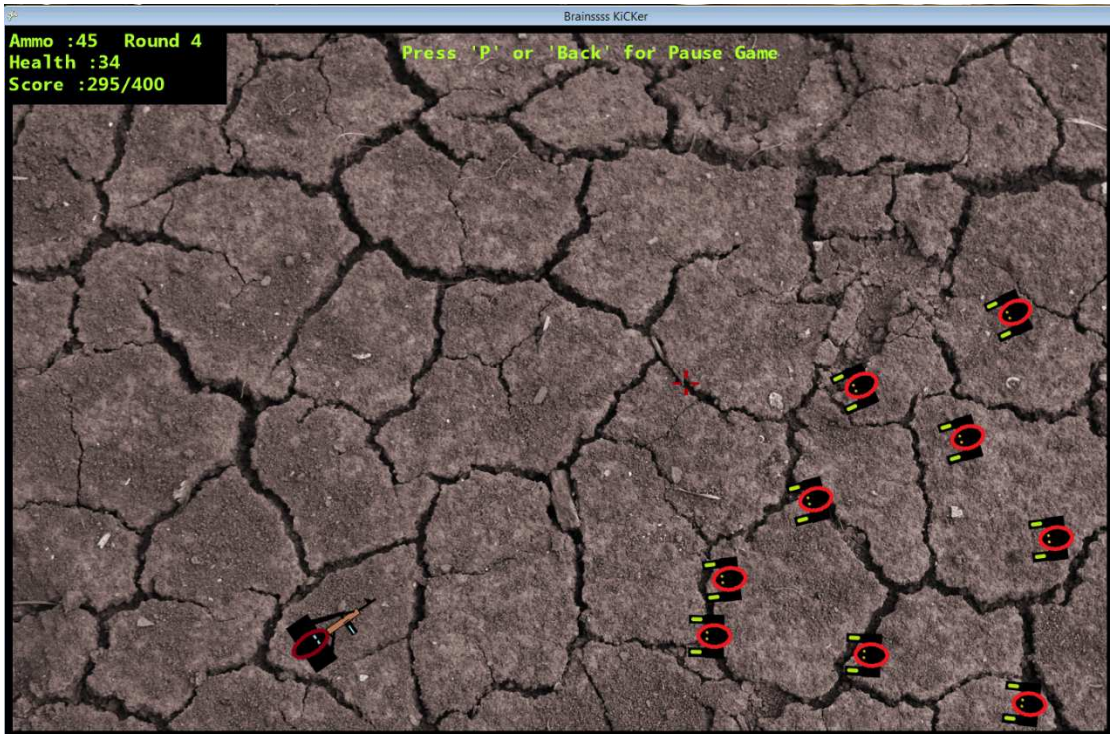
Το περιβαλλον του παιχνιδιου[3]

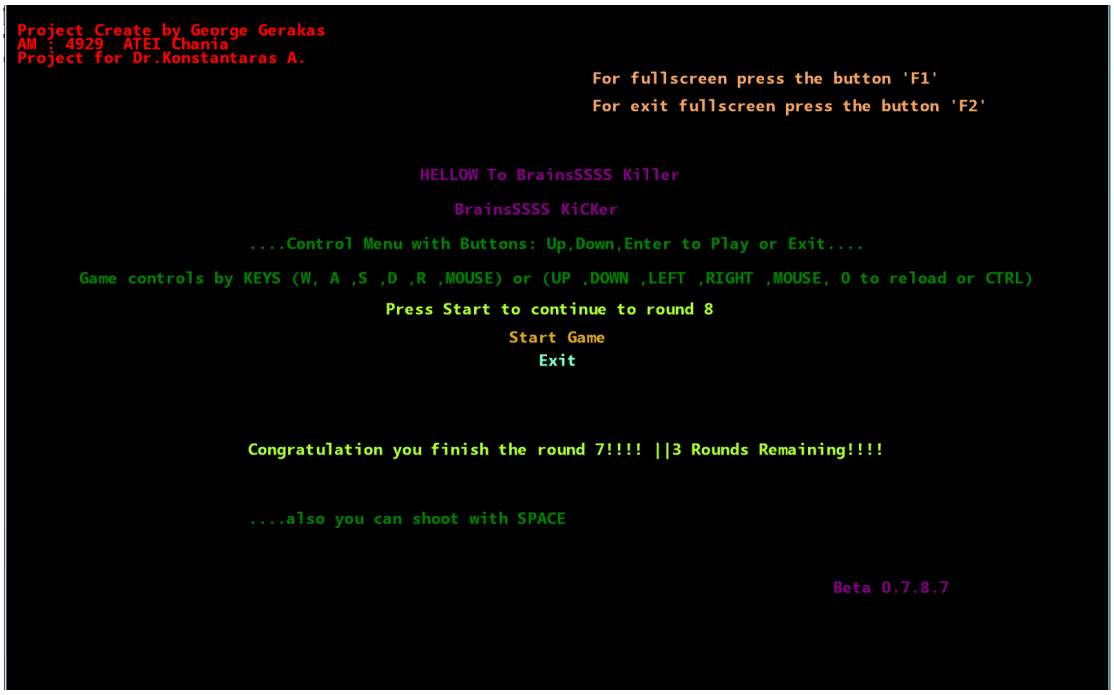









Brainssss KiCKer
Ammo :45 Round 4
Health :34
Score :295/400
Press 'P' or 'Back' for Pause Game

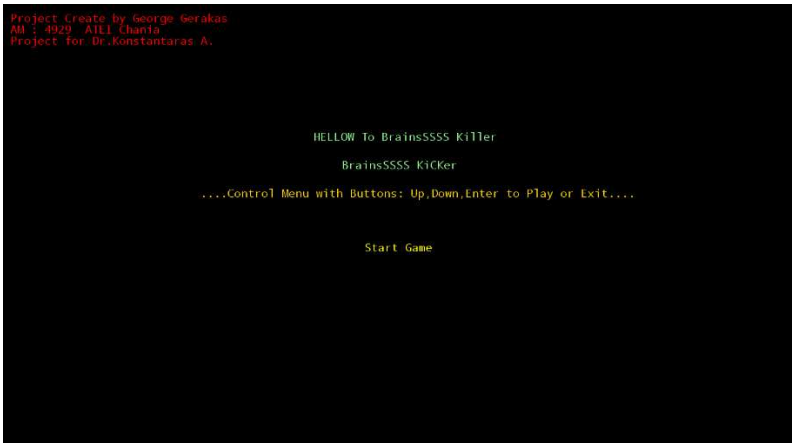

Project Create by George Gerakas
AM : 4929 ATEI Chania
Project for Dr.Konstantaras A.
For fullscreen press the button 'F1'
For exit fullscreen press the button 'F2'
HELLOW To BrainsSSSS Killer
BrainsSSSS KiCKer
....Control Menu with Buttons: Up,Down,Enter to Play or Exit....
Game controls by KEYS (W, A ,S ,D ,R ,MOUSE) or (UP ,DOWN ,LEFT ,RIGHT ,MOUSE, 0 to reload or CTRL)
Press Start to continue to round 8
Start Game
Exit
Congratulation you finish the round 7!!!! ||3 Rounds Remaining!!!!
....also you can shoot with SPACE
Beta 0.7.8.7

Στάδια βελτίωσης του παιχνιδιού

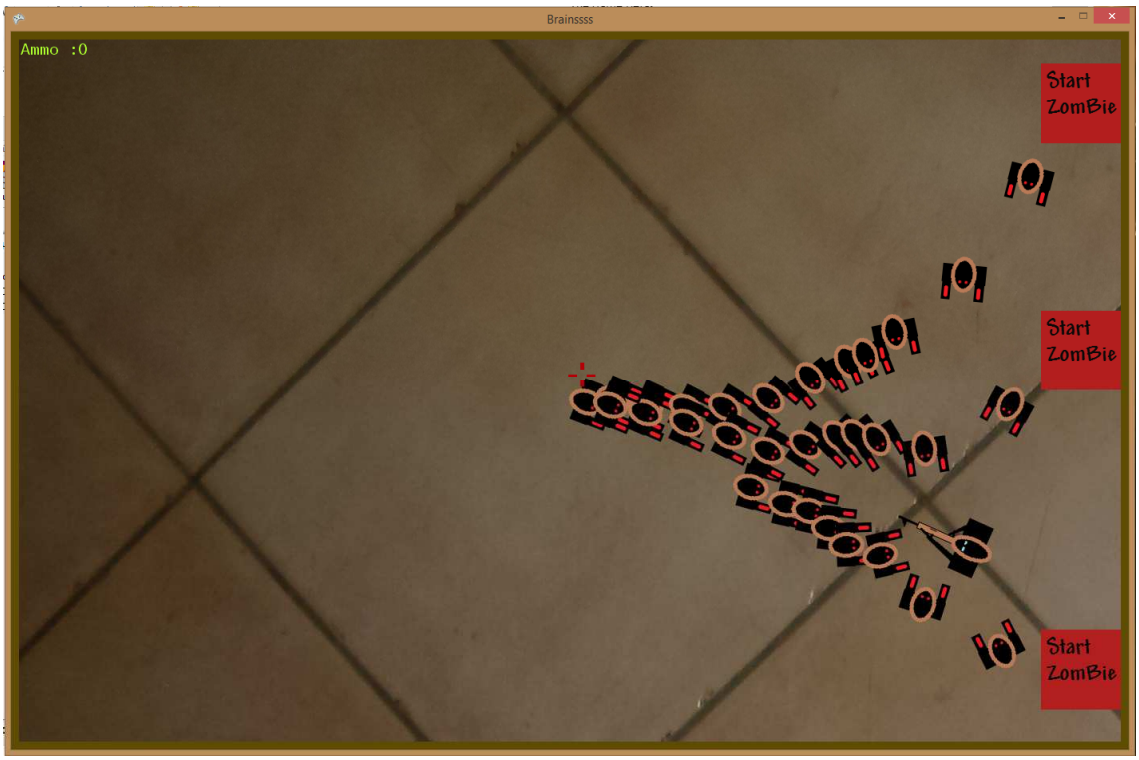
Ενδιάμεσα απο αυτα τα Beta υπηρξαν και αλλα με μικροαλλαγες

Ημερομηνια:5/7/2014 Beta 0.1.3.2

Πολυ λιτο μενου χωρις πληροφοριες

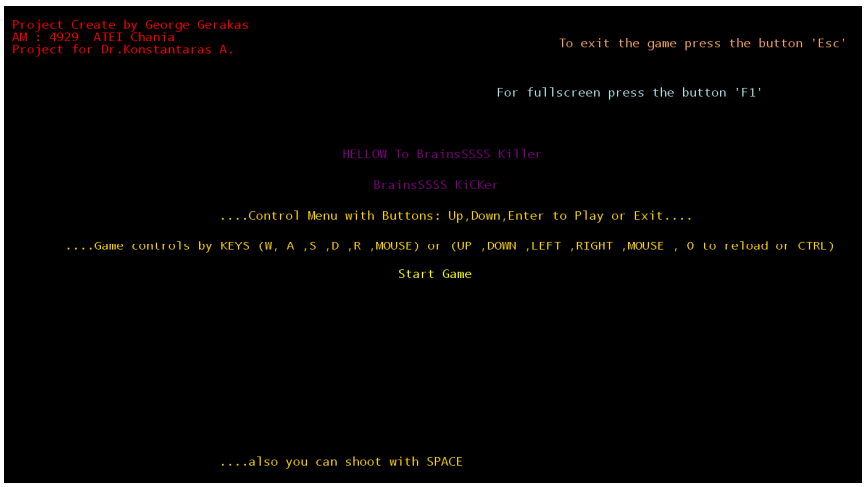


Τα γραφικά του παιχνιδιού είναι πολύ απλά και το παιχνίδι δεν έχει νόημα γιατι δεν τελιώνει ποτέ και ο παίκτης δεν χάνει ζωή οταν τον χτυπάνε τα ζόμπι. Επίσης οταν τελειώσουν οι σφαίρες, το όπλο δεν οπλίζει αυτόματα αλλα πρέπει να πατήσεις το πλήκτρο (R).



Ημερομηνια:25/7/2014 Beta 0.5.0.6

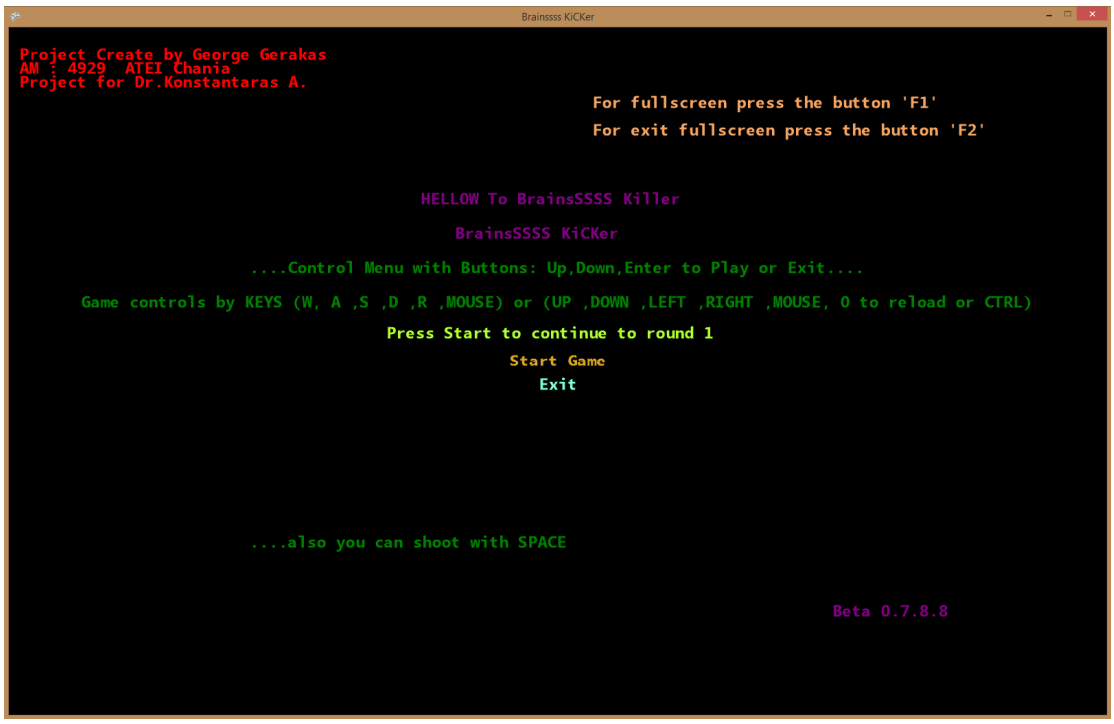
Το μενού έχει περισσότερες πληροφορίες και οδηγίες.



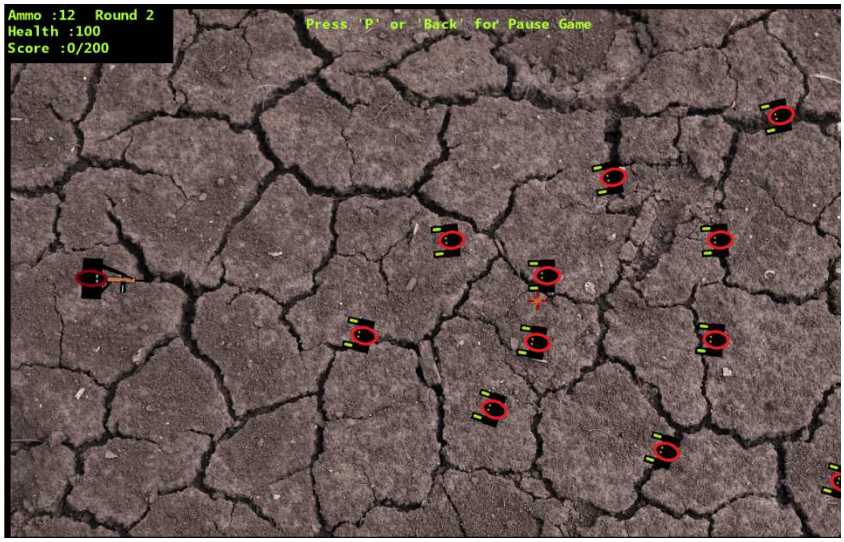
Τα γραφικά του παιχνιδιού αλλάζουν και το παιχνίδι αποκτά νόημα γιατι οταν τελιώσει ζωή ο παικτης χανει. Επίσης οταν τελειώσουν οι σφαίρες, το όπλο οπλίζει αυτόματα και ο χρηστης μπορεί να κάνει διάλειμμα απο το παιχνίδι(pause).

Ημερομηνια:22/9/2014 Beta 0.7.8.8

Η τελική μορφή του μενού του παιχνιδιού που τα εχει ολα!

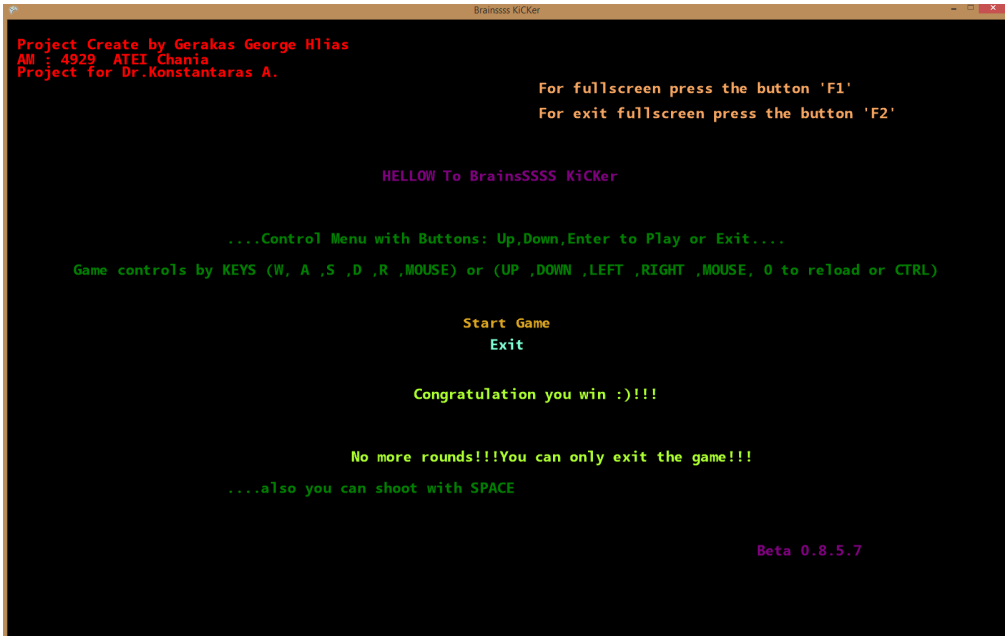


Τα γραφικά του παιχνιδιού αλλάζουν και γινονται πιο ενδιαφεροντα. Το παιχνίδι αποκτά νόημα γιατι ο παικτης χάνει και υπαρχουν 10 γυροι δυσκολιας στο παιχνιδι. Επίσης οταν τελειώσουν οι σφαίρες, το όπλο οπλίζει αυτόματα και ο χρηστης μπορεί να κάνει διάλειμμα απο το παιχνίδι(pause).Το μονο προβλημα τωρα ειναι οτι οι εχθροι που αφηνει ο χρηστης ζωντανους οταν αλλαζει ο γυρος ειναι ζωντανοι στον επομενο γυρο, ετσι με το που αρχισει ο επομενος γυρος οι εχθροι ειναι εκει που μειναν στον προηγουμενο γυρο και συνεχιζουν να τον κυνηγουν.

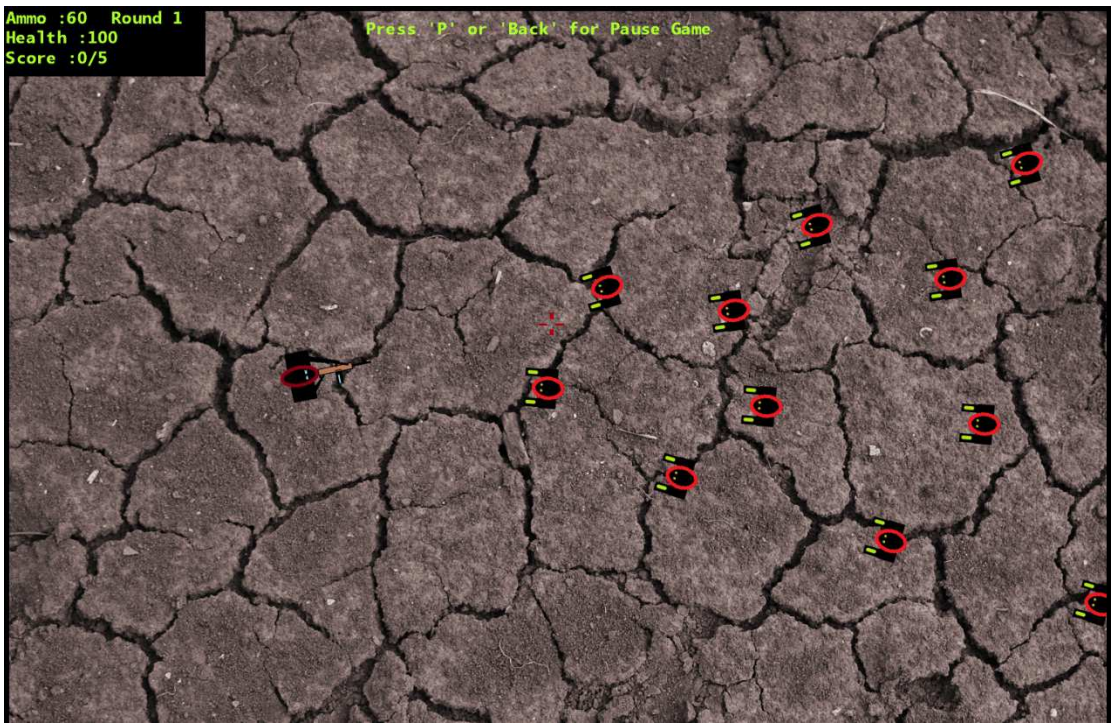


Ημερομηνια:3/11/2014 Beta 0.8.5.7

Το μενού του παιχνιδιού παραμένει ίδιο.Οταν ο παικτης κερδισει και τον τελευταιο γυρο το παιχνιδι γυρναει στο μενου με τα παρακατω μηνυματα και ο χρηστης το μονο που μπορει να κανει ειναι να πατησει το κουμπι exit.



Το παιχνίδι παραμενει ιδιο και ακομα διορθωνεται το παραπανω προβλημα που αναφερθηκε.



Βιβλιογραφια

1) Wikipedia.

2) Microsoft helper.

3) Microsoft Developer Network.

4) Learn Programming Now! (Συγγραφέας: Rob Miles)

5) XNA 4.0 Game Development by Example(Συγγραφέας: Kurt Jaegers)

6) How to Make a Game in XNA[εκπαιδευτικο βιντεο](Salad Raider)

7) Learning XNA 4.0(Συγγραφέας: Aaron Reed)

8) Introduction to XNA(Συγγραφέας: Paul Rodriguez, Victor Lin, Anthony Balmeo, Behnood Marvazi)

9) Άρθρο για το collision του Peter Kuhn "Mister Goodcat"

10) Function X press C# Aplication design(rectangle diagram)

11) Εισαγωγή στη C# www.studentguru.gr/(Δημήτρης Γκανάτσιος)