Race

משחק מכוניות בעולם פתוח

שם: אור מצליח

תעודת זהות: 212105209

שם המורה: יהודה אור

בית ספר: תיכון הכפר הירוק ע"ש לוי אשכול

שם הפרויקט: Race – משחק מכוניות בעולם פתוח

כיתה: י"ב\2

תעודת זהות: 212105209

תוכן עניינים

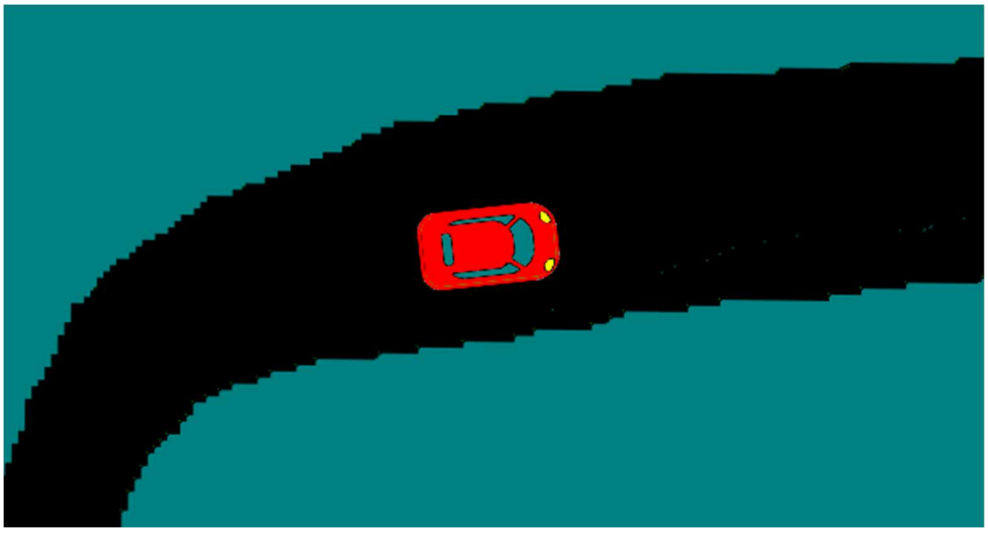
רפלקציה

במהלך הלימודים השנה במדעי המחשב למדתי המון דברים חדשים ובעיקר שכללתי את יכולות התכנות שלי. מאוד נהניתי לעבוד על הפרויקט השנה, נתקלתי בקשיים במהלך ההכנה של הפרויקט וגם הצלחתי לפתור אותם. למדתי כיצד לתעד את הקוד בצורה מסודרת ומובנת יותר ממה שידעתי לפני, וכך הצלחתי להפוך את חווית התכנות לכיפית יותר ומסודרת יותר.

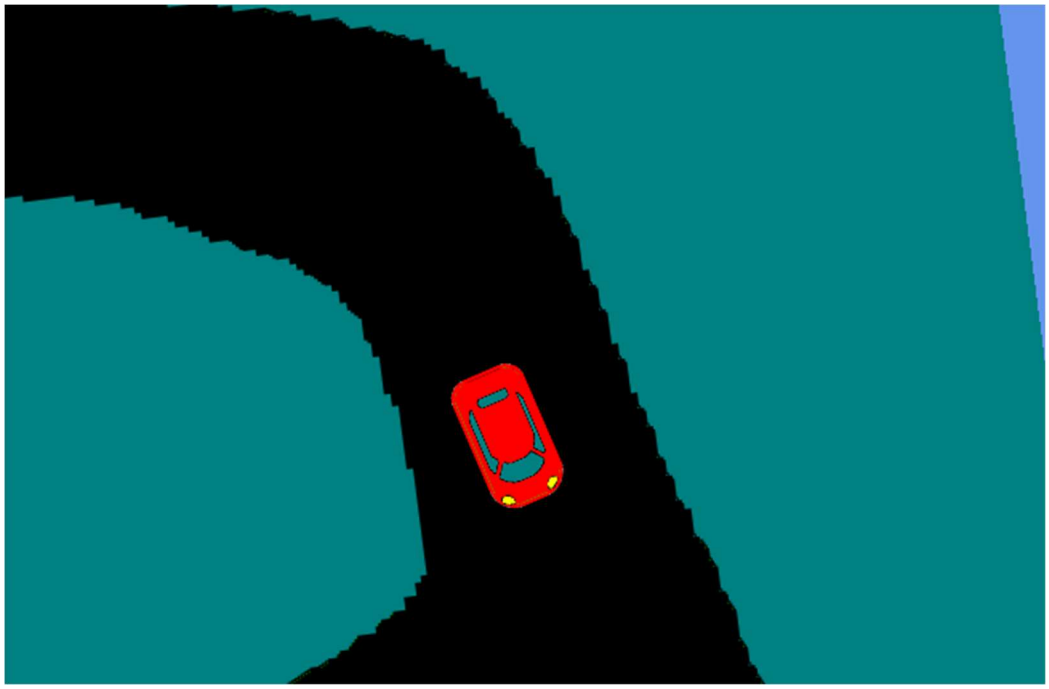
במהלך הכנת הפרויקט, למדתי כיצד אפשר ליצור משחק עולם פתוח, שיוצא מגבולות המסך בעזרת מצלמה שעוקבת אחרי אובייקט לבחירתי. בנוסף, למדתי כיצד ליצור מערכת שמזהה התנגשות בעזרת חלוקה של הרקע לאזורים לפי צבעים.

אחד הקשיים הגדולים שהיו לי במהלך הפרויקט היה הבחירה של הנושא לפרויקט. לאחר הרבה חשיבה החלטתי להכין משחק עם מכונית, מכיוון שאני מאוד אוהב משחקי מירוצים.

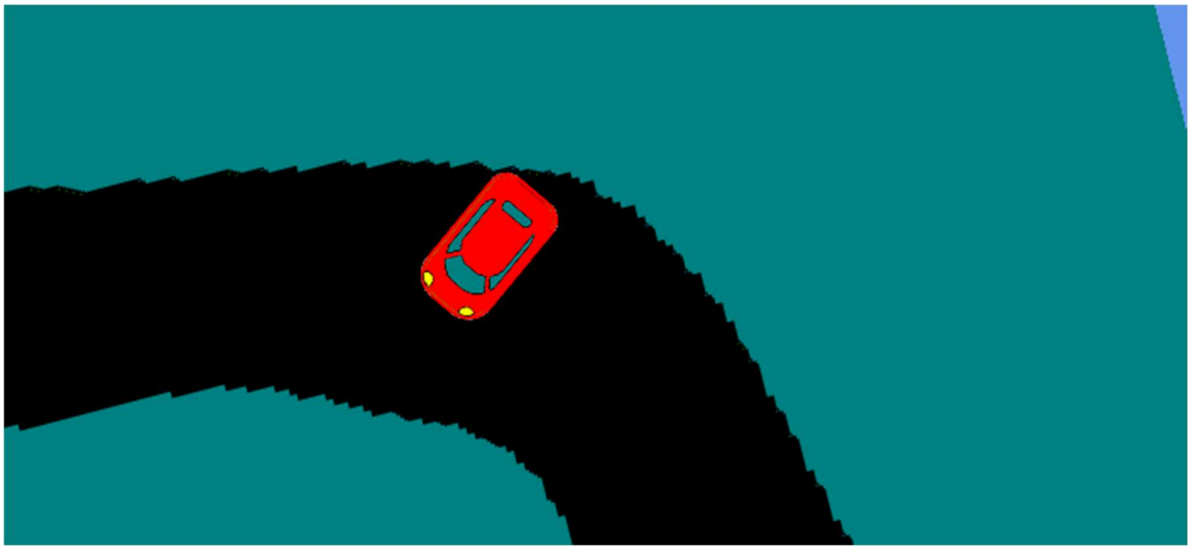
תמונות



מצלמה שעוקבת אחרי המכונית

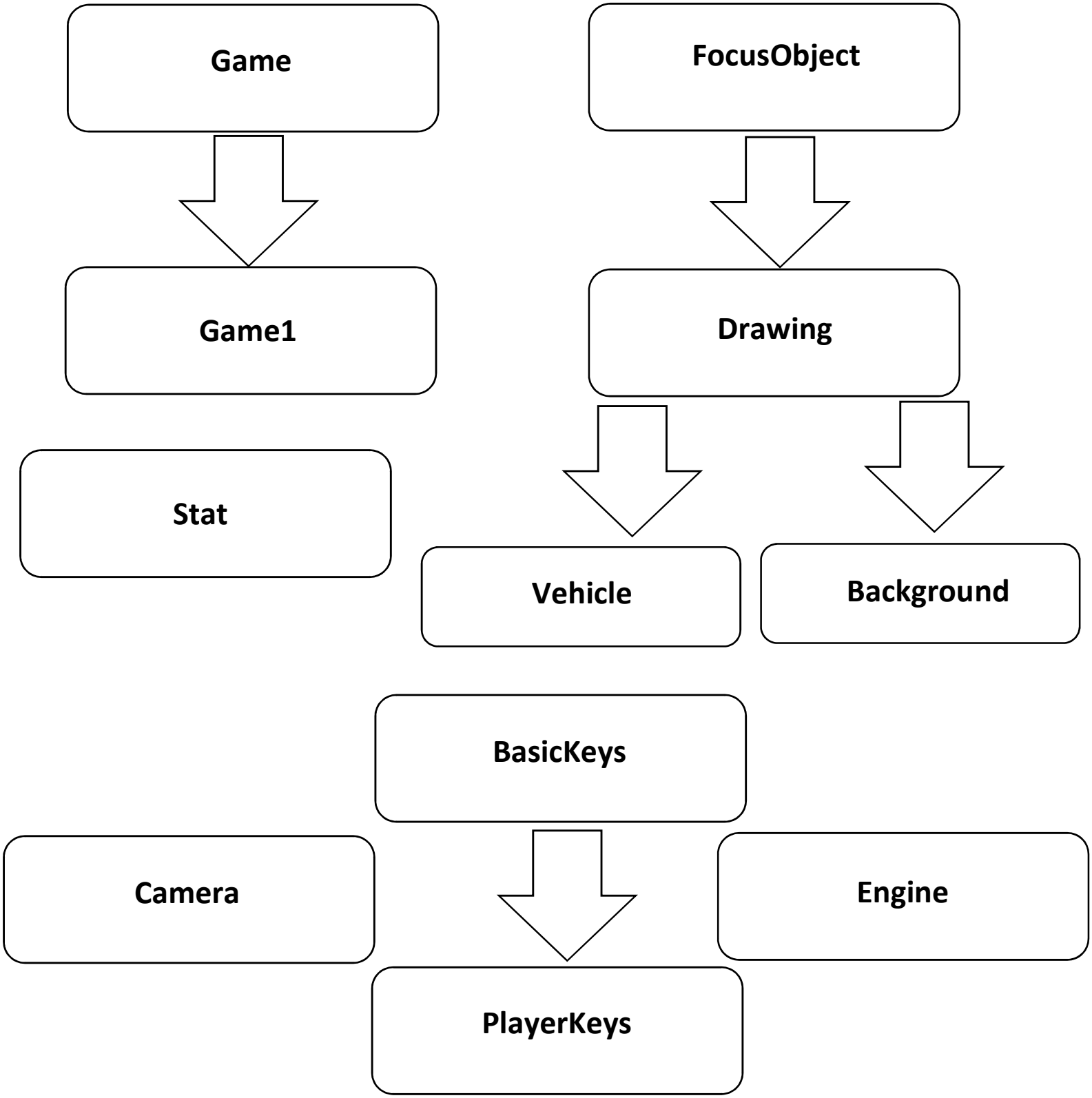


המצלמה מסוגלת גם לשנות את זווית הצפייה בהתאם לזווית שבה נמצאת המכונית.



במשחק מוטמעת מערכת אשר מונעת מהנהג לצאת מגבולות הכביש. כאשר מנסה השחקן לצאת מגבולות הכביש, המכונית מוחזרת אחורה ומהירות המכונית יורדת ל-0.

קשר בין מחלקות - UML



חלק תיאורטי

שפת #C:

#C (מבטאים סי שארפ) היא שפת תכנות עילית, מונחת עצמים שפותחה על ידי מיקרוסופט בשנת 2000. שפת תכנות עילית הינה שפה שמיועדת לשימוש על ידי מתכנתים אנושיים, וכתובה בשפה יותר מובנת לאדם, מאשר שפת תכנות נמוכה כמו אסמבלי שכתיבתה יותר קרובה לשפת מכונה, דבר שהרבה פחות קרוב לשפת אדם ודורש הבנה גדולה יותר של איך עובד המחשב. בשפה זאת השתמשתי לכתיבת הפרויקט. מכיוון שהשפה הזאת מאוד דומה לשפת Java שלמדנו לבגרות בי"א, היה פשוט יחסית לכתוב בשפה זאת מכיוון שבחלק גדול מהמקרים הכתיבה של הקוד דומה מאוד או אפילו זהה לכתיבה של קוד זהה בשפת Java.

Visual Studio:

Visual Studio היא סביבת פיתוח שפותחה על ידי מיקרוסופט. ניתן לתכנת ב- Visual Studio במספר רב של שפות כגון ++C, VB.NET, #C (השפה שבה נכתב הפרויקט) ועוד. Visual Studio מאפשרת פיתוח למעבדים שונים ומערכות הפעלה שונות, כולל מחשבים שולחניים וטלפונים. בנוסף, ישנה תמיכה בתוספים שיכולים להוסיף תמיכה בשפות חדשות, וגם תוספים כמו Monogame שמאפשרים שימוש קל יותר ב- DirectX. הגרסה הראשונית של התוכנה שוחררה ב-1977, בעוד ש- Visual Studio 2017 (הגרסה שהשתמשתי בה לכתיבת הפרויקט שוחררה ב-7.3.2017.

מחלקה:

המסגרת הבסיסית בשפות מונחות עצמים. מחלקה היא אוסף של משתנים, פונקציות ומשתנים, שמאוחדים למבנה אחד ופועלים ביחד. בנוסף, אפשר לבנות למחלקה בנאי (Constructor) שמאפשר ליצור עצם מסוג חדש, עם תכונות שקובע המתכנת. למשל, למחלקה בפרויקט שלי שנקראת Engine, יש בנאי שדורש ערך של מהירות מקסימלית, ומהירות האצה. לאחר השימוש בבנאי נקבעת המהירות הנוכחית של המנוע ל-0.

MonoGame:

MonoGame הינו תוסף קוד פתוח ל Visual Studio שמאפשר שימוש ב-DirectX בעזרת השפה C#. בעזרת התוסף מתאפשרת גישה קלה יותר לציור על המסך, אך לעומת מנוע כמו Unity, MonoGame אינו מהווה תחליף לפיזיקה של המשחק, ומעביר את האחריות לכתיבה שלה למתכנת.

פולימורפיזם:

תכונה של שפות תכנות המאפשרת עבודה אחידה על אובייקטים מסוגים שונים. בעזרת תכונה זו ניתן לכתוב אלגוריתמים וממשקים אחידים שיאפשרו למתכנת לעבוד עם אובייקטים שונים למרות שונותם בלי לכתוב כמות גדולה של קוד לכל מקרה ספציפי. בפרויקט שלי השתמשתי בפולימורפיזם על מנת לבנות את ממשק המצלמה, ובכך המצלמה יכולה להתרכז על אובייקטים מסוגים שונים בעזרת הממשק שדורש מיקום וזווית סיבוב. בכך במקום שהמצלמה תתרכז על אובייקט מסוג כלי רכב, יכולה המצלמה להתרכז גם על אובייקטים אחרים.

ירושה:

ירושה היא פעולה המאפשרת למחלקה חדשה לקבל את כל התכונות של המחלקה שהיא יורשת ממנה. ירושה בעצם יוצרת סוג של העתק של המחלקה הראשונה בלי הצורך של כתיבה מחדש של הקוד, בנוסף, אפשר להוסיף למחלקה היורשת פעולות חדשות ומימושים חדשים, שיאפשרו שימושים רבים יותר מאשר מה שאפשרי במחלקה המקורית. במהלך הפרויקט השתמשתי בירושה מספר פעמים, דבר שניתן לראות בתרשים ה-UML שמצורף למעלה.

אירועים:

אירועים (נקראים באנגלית Events), מאפשרים הפעלת פעולות שונות, ממחלקות שונות, על ידי קריאה אחידה. על מנת להפעיל פעולה מסוימת ממחלקה על ידי הקריאה, נדרש לרשום את הפעולה לקריאה. על ידי ההרשמה הזאת, ידע המחשב בכל פעם שיש קריאה, להפעיל את הפעולות שרשומות, ובכך אפשר להפעיל מספר פעולות גדול, על ידי קריאה אחת ויחידה, שתפעיל את כל הפעולות שרשומות אליה.

```
public Vehicle(Texture2D tex, Vector2 position, Vector2 Origin, Engine engine, bool events)
    : base(tex, position, Color.White, 0.0f, new Vector2(1, 1), 0.0f)
{
    this.engine = engine;
    this.Velocity = Velocity;
    if (events) Game1.EVENT_UPDATE += this.Update;
}
#endregion

#region Update
public void Update(GameTime gameTime) //the update function for the vehicle
{
    deltaRotation = 0;
    if(basicKeys.Right() && Math.Abs(engine.Speed) > 0.3f)
    {
        deltaRotation += 0.03f;
    }
    if (basicKeys.Left() && Math.Abs(engine.Speed) > 0.3f)
    {
        deltaRotation -= 0.03f;
    }
    rotation += deltaRotation;
    Vector2 direction = Vector2.Transform(Vector2.UnitX, Matrix.CreateRotationZ(rotation));
    Velocity = engine.Speed * direction;
    if (!(Stat.background[position.X + Velocity.X, position.Y + Velocity.Y] == Background.BackgroundType.Road))
```

הרשמה של פעולת Update במחלקת Vehicle, לאירוע EVENT_UPDATE ממחלקת Game1. ההרשמה מאפשרת לפעולה לפעול בכל פעם שמתבצעת קריאה לאירוע.

```
protected override void Update(GameTime gameTime)
{
    if (EVENT_UPDATE != null) EVENT_UPDATE(gameTime);
    cam.UpdateMatri();
    // TODO: Add your update logic here

    base.Update(gameTime);
}
```

קריאה לאירוע EVENT_UPDATE בפעולת Update במחלקת Game1 הקריאה הזאת מאפשרת לאירוע לפעול בכל פעם שפעולת פעולת ה-Update במחלקה.

מחלקה אבסטרקטית:

מחלקה אבסטרקטית היא מחלקה שמאפשרת לבנות פעולות ללא מימוש, על מנת לבנות מחלקות עם פעולות זהות, אך עם מימושים שונים. בעזרת ירושה ניתן לרשת את הפעולות שאינם ממומשות למחלקה אחרת ולממש אותם, כל ירושה יכולה לאפשר מימוש בצורה שונה. דוגמה למחלקה כזאת בפרויקט שלי היא מחלקת BasicKeys שכוללת בתוכה ארבעה משתנים בוליאנים ללא מימוש. הסיבה שבחרתי לעשות מחלקה כזאת היא מכיוון ששחקן שמשחק יהפוך את המשתנים לאמת בעזרת לחיצת כפתור, לעומת שחקן שנשלט על ידי המחשב, שאינו יכול ללחוץ על כפתורים במקלדת. למרות שעדיין לא הספקתי לעבוד על בוטים, המחלקה האבסטרקטית תאפשר לי לעבוד על מימוש של תזוזה של שחקנים שנשלטים על ידי המחשב, שתהיה שונה מהמימוש של דמות שנשלטת על ידי שחקן שלוחץ על כפתורים במקלדת.

```
namespace OrProject
{
    #region BasicKeys
    abstract class BasicKeys
    {
        public abstract bool Up();
        public abstract bool Down();
        public abstract bool Right();
        public abstract bool Left();
    }
```

בתמונה ניתן לראות את המחלקה האסטרקטית BasicKeys,
שבה יש ארבעה משתנים בולאנים לא ממומשים.
המשתנים מוגדרים כאבסטרקטיים על מנת ליידע שהמשתנים
האלו לא עוברים מימוש כרגע, אלא יעברו מימוש בעתיד
על ידי מחלקות אחרות שירשו אותם (PlayerKeys).

```
class PlayerKeys : BasicKeys
{
    private Keys up, down, right, left;
    #endregion

    #region Constructor
    /// <summary> A constructor used to determine the keys used
    public PlayerKeys(Keys up, Keys down, Keys right, Keys left)
    {
        this.up = up;
        this.down = down;
        this.right = right;
        this.left = left;
    }
    #endregion

    #region up
    public override bool Up()
    {
        return Keyboard.GetState().IsKeyDown(up);
    }
    #endregion
```

בתמונה ניתן לראות את מחלקת PlayerKeys שיורשת את המחלקה האבסטרקטית BasicKeys, בנוסף המחלקה מבצעת מימוש לבוליאנים השונים מהמחלקה שהיא יורשת ממנה, למשל, בבבוליאן Up מבצעת המחלקה מימוש בעזרת בדיקה האם הכפתור שנשלח בזמן בניית האובייקט שנשמר בשם up נלחץ. אם כן המחלקה מחזירה אמת. אם לא המחלקה מחזירה שקר.

חלק מעשי

השנה במסגרת הלימודים שלנו במדעי המחשב, קיבלנו משימה להכין פרויקט שיהווה 5 יחידות לימוד נוספות על ה5 הקודמות שסיימנו בכיתה י"א. במהלך השנה עבדנו על הפרויקט גם במסגרת השיעור ביחד עם יהודה שהסביר לנו שיטות שונות, ודברים שונים שאפשר להוסיף לפרויקט שלנו, וגם בבית, כעבודה עצמית.

במהלך הכנת הפרויקט נתקלתי במספר התלבטויות. התלבטות אחת שהייתה לי היא בחירת נושא לפרויקט. ידעתי שרציתי להכין משחק כלשהו לפרויקט, אבל לא ידעתי בדיוק מה יהיה הנושא של המשחק. לבסוף החלטתי לבחור לעשות משחק מכוניות, מכיוון שאני מאוד אוהב מכוניות ומשחקי מרוצים.

התלבטות נוספת שהייתה לי היא כיצד ליצור את מערכת ההתנגשות. בהתחלה לא ידעתי איך לעשות את זה, אבל אחרי הרבה מחשבה, חשבתי על האפשרות לקביעת ההתנגשות על פי צבע. זאת אומרת, לכל משטח יהיה צבע שונה ואחיד וכך אוכל למפות את הרקע ולחלק אותו למשטחים שונים. כך למשל, כל פיקסל בצבע שחור יהווה כביש, שעליו המכונית תוכל לנסוע בחופשיות. פיקסל בצבע ירוק יהיה דשא, ולכן אמנע מהמכונית לצאת מגבולות הכביש על ידי בדיקה של מיקום המכונית, והאם היא נמצאת על פיקסל בצבע ירוק. אם כן אדאג לעצור את המכונית ולהחזיר אותה אחורה

לגבולות הכביש. בעזרת המנגנון הזה הצלחתי לבנות מערכת התנגשות שדואגת לשמור על המכוניות במשחק בגבולות הכביש.

```
public Background(Texture2D tex)
    :base(tex, Vector2.One)
{
    background = new BackgroundType[tex.Width, tex.Height];
    Color[] Colors = new Color[tex.Width * tex.Height];
    tex.GetData<Color>(Colors);
    for (int w = 0;  w < tex.Width;  w++)
        /*this function puts the default background type as edge, if the pixel is black it sets it to road
        and if green to grass*/
    {
        for (int h = 0; h < tex.Height; h++)
        {
            background[w, h] = BackgroundType.Edge;
            if(Colors[w + h * tex.Width] == Color.Black)
            {
                background[w, h] = BackgroundType.Road;
            }
            if(Colors[w + h * tex.Width] == Color.Green)
            {
                background[w, h] = BackgroundType.Grass;
            }
        }

    }
}
```

הקוד שמאפשר מיפוי של הרקע למשטחים סוגים שונים, פיקסלים בצבע שחור נקבעים כ-Road, פיקסלים בצבע ירוק כ-Grass, ושאר הפיקסלים נקבעים כ-Edge

בנוסף, התלבטתי האם לבנות את המשחק כמשחק עולם פתוח, שיאפשר לשחקן לצאת מגבולות המסך, ולבנות בנוסף מצלמה שתעקוב אחרי השחקן. לאחר מחשבה רבה, החלטתי שכן כדאי לבנות מצלמה שתעקוב אחרי השחקן, מכיוון שדבר זה מוסיף לקושי של המשחק, מכיוון שהשחקן לא יכול לראות את כל המסלול, ויהיה לו יותר קשה להעריך מראש את הפניות שהוא צריך לבצע. בחרתי גם לאפשר למצלמה להתרכז על אובייקטים מסוגים שונים, כדי שיתאפשר ביתר קלות להשתמש במצלמה הזאת גם כדי להתרכז על דברים שונים מהמכונית של השחקן, אם אחליט שארצה בעתיד.

```
public Camera(Viewport vp, FocusObject focus, Vector2 zooming)
{
    this.focus = focus;
    this.zooming = zooming;
    this.vp = vp;
    Position = new Vector2(0,0);
}
#endregion

#region UpdateMatri
public void UpdateMatri()
    /* A function used to apply a Matrix that first moves the object to 0,0 then applies a rotation,
       then applies the scale, and then moves the camera so the object will be in the middle */
{
    matri = Matrix.CreateTranslation(-Position.X, -Position.Y, 0) *
        Matrix.CreateRotationZ(-focus.rotation * 0.1f) *
        Matrix.CreateScale(this.zooming.X, this.zooming.Y, 1f) *
        Matrix.CreateTranslation(vp.Width/2, vp.Height/2, 0);
    Position = Vector2.Lerp(focus.position, Position, 0.5f);
}
#endregion
```

הקוד שמאפשר למצלמה לעקוב אחרי עצמים מסוגים שונים. על מנת לגרום למצלמה להתרכז על האובייקט, המצלמה מפעילה מטריצה, שבתחילה מביאה את האובייקט לנקודת ה-(0,0), לאחר מכן מפעילה את הסיבוב, על מנת לסובב את המצלמה, לאחר מכן, הגודל של התמונה מוכפל ב-Scale, על מנת לאפשר זום על השחקן במידת הצורך, ולבסוף, מופעלת עוד הזזה, שמזיזה את האובייקט למרכז המסך.

התלבטות נוספת הייתה האם לאפשר למצלמה לשנות זווית בעקבות שינוי זווית של האובייקט שהיא מתרכזת עליו, והחלטתי שכן לאפשר למצלמה לעשות את זה על מנת לאפשר לשחקן זוויות צפייה נוחות יותר, שיאפשרו ביתר קלות לשחקן לעקוב אחרי המכונית ולהעריך בקלות רבה יותר את המשך פעולותיו על מנת להתקדם במסלול.

קוד מתועד

מחלקת Program:

המחלקה הראשית של הפרויקט, מטרתה ליצור אובייקט מסוג Game, על מנת שיהיה אפשר להתשמש בו להרצת המשחק.

```
using System;

namespace OrProject
{
#if WINDOWS || LINUX
    /// <summary>
    /// The main class.
    /// </summary>
    public static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            using (var game = new Game1())
                game.Run();
        }
    }
#endif
}
```

מחלקת Game1:

המחלקה המרכזית של הפרויקט, תפקידה לרכז מספר רב של פעולות פנימיות שמשמשות לתפקידים שונים:

()Initialize: פעולה שרצה פעם אחת עם הפעלת הפרויקט, תפקידה לאפשר שינוי של תכונות המשחק (כגון גודל המסך או האם להראות את סמן העכבר). מכיוון שהפעולה רצה רק פעם אחת, ניתן להשתמש בפעולה הזו רק לצורך של שינוי של תכונות שלא צריך לשנות בעתיד.

()LoadContent: פעולה שרצה פעם אחת עם הפעלת הפרויקט, תפקידה לשמש מקום לריכוז המשתנים, כמו טקסטורות, שמשמשים במהלך המשחק.

()Update: פעולה שרצה 60 פעמים בשנייה, ותפקידה לרכז את כל השינויים והעדכונים שצריך המתכנת.

()Draw: רצה על כל 60 קריאות של פעולת ה-Update. תפקידה לרכז את כל הפעולות לציור על המסך שהמתכנת כתב. ניתן לצייר על המסך בעזרת המשתנה SpriteBatch, ששייך למחלקה, ויש לו גישה לכרטיס המסך, שבעזרתו אפשר לצייר על המסך.

```
#region data
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
#endregion

namespace OrProject
{
    /// <summary>
    /// This is the main type for your game.
    /// </summary>
```

```
public class Game1 : Game
  {
    #region data
    public static DlgtUpdate EVENT_UPDATE;
    public static DlgtDraw EVENT_DRAW;
    GraphicsDeviceManager graphics;
    SpriteBatch spriteBatch;
    Texture2D maptex;
    Drawing car;
    Texture2D cartex;
    Engine engine;
    Vehicle vehicle;
    Camera cam;
    #endregion

    public Game1()
    {
      graphics = new GraphicsDeviceManager(this);
      Content.RootDirectory = "Content";
    }

    /// <summary>
    /// Allows the game to perform any initialization it needs to before starting to run.
    /// This is where it can query for any required services and load any non-graphic
    /// related content.  Calling base.Initialize will enumerate through any
      components
    /// and initialize them as well.
    /// </summary>
    protected override void Initialize()
    {
      // TODO: Add your initialization logic here

      base.Initialize();
    }
```

```
/// <summary>
/// LoadContent will be called once per game and is the place to load
/// all of your content.
/// </summary>



protected override void LoadContent()
{
    // Create a new SpriteBatch, which can be used to draw textures.
    spriteBatch = new SpriteBatch(GraphicsDevice);
    Stat.Initialize(Content, spriteBatch, GraphicsDevice);
    maptex = Content.Load<Texture2D>("Track");
    cartex = Content.Load<Texture2D>("car");
    Stat.background = new Background(maptex);
    engine = new Engine(8f, 0.5f);
    vehicle = new Vehicle(cartex, new Vector2(300, 260), new Vector2(0,0),
    engine, true);
    vehicle.basicKeys = new PlayerKeys(Keys.Up, Keys.Down, Keys.Right,
    Keys.Left);
    vehicle.scale = Vector2.One * 0.4f;
    vehicle.makeTransparentColor(Color.White);
    cam = new Camera (new Viewport(GraphicsDevice.Viewport.Bounds),
    vehicle, new Vector2(1f));
    // TODO: use this.Content to load your game content here
}

/// <summary>
/// UnloadContent will be called once per game and is the place to unload
/// game-specific content.
/// </summary>
```

```
protected override void UnloadContent()
    {
        // TODO: Unload any non ContentManager content here
    }

    /// <summary>
    /// Allows the game to run logic such as updating the world,
    /// checking for collisions, gathering input, and playing audio.
    /// </summary>
    /// <param name="gameTime">Provides a snapshot of timing values.</param>

    protected override void Update(GameTime gameTime)
    {
        if (EVENT_UPDATE != null) EVENT_UPDATE(gameTime);
        cam.UpdateMatri();
        // TODO: Add your update logic here

        base.Update(gameTime);
    }

    /// <summary>
    /// This is called when the game should draw itself.
    /// </summary>
    /// <param name="gameTime">Provides a snapshot of timing values.</param>
```

```
    protected override void Draw(GameTime gameTime)
     {
        GraphicsDevice.Clear(Color.CornflowerBlue);

        spriteBatch.Begin(SpriteSortMode.Deferred, BlendState.AlphaBlend, null,
null, null,
            null, cam.matri);
        if (EVENT_DRAW != null) EVENT_DRAW();
        Stat.background.Draw();
        vehicle.Draw();
        spriteBatch.End();

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

מחלקת FocusObject:

המחלקה הזו נועדה לאפשר לרכז את המצלמה על אובייקטים מסוגים בעזרת שימוש בפולימורפיזם. המחלקה דורשת משתנה של מיקום על מנת לדעת איפה האובייקט, ומשתנה שמודד את הסיבוב, על מנת שיהיה ניתן לסובב את המצלמה.

```
#region using
using Microsoft.Xna.Framework;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
#endregion

namespace OrProject
{
   #region FocusObject
   interface FocusObject
  //An interface made for being able to focus the camera on an object of any type
      {
         Vector2 position { get; }
         float rotation { get; }
      }
   #endregion
}
```

מחלקת Drawing:

מחלקת Drawing היא מחלקה שנועדה לאפשר לכותב ליצור אובייקט עם פרמטרים, שיאפשר לצייר טקסטורה על המסך. למחלקה שני בנאים, אחד שנועד בשביל הרקע, שדורש טקסטורה, ווקטור שאיתו הוא משנה את גודלה. הבנאי השני דורש יותר פרמטרים ונועד לשאר הדברים שצריך לצייר על המסך. בנוסף למחלקה פעולה שנקראת makeTranparentColor שמקבלת פרמטר של צבע. תפקידה לעבור על כל הטקסטורה ולמצוא פיסקלים בצבע שקיבלה, ולהפוך אותם לצבע שקוף.

```
#region using
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
#endregion

namespace OrProject
{

    class Drawing:FocusObject
    {
        #region Data
        protected Texture2D tex;
        protected Rectangle? sourceRec;
        protected Vector2 origin;
        public float rotation { get; set; }
        public Vector2 position { get; set; }
        protected Color color;
        public Vector2 scale;
        SpriteEffects effect;
        float layerDepth;
        #endregion
```

```
#region Constructors
/// <summary>
/// Constructor for a drawing
/// </summary>
/// <param name="tex">the texture of the drawing</param>
/// <param name="position">the position of the drawing compared to
0,0</param>
/// <param name="color">Used to change the color of the drawing, Color.White
for no change</param>
/// <param name="rotation">Used to rotate the drawing</param>
/// <param name="scale">used to change the size of the drawing</param>
/// <param name="layerDepth">Used to determine which drawing would be on
top, 1 is to be on top</param>

public Drawing (Texture2D tex, Vector2 position, Color color, float rotation,
    Vector2 scale, float layerDepth = 0) //constructor for a drawing
{
    this.tex = tex;
    this.position = position;
    this.color = color;
    this.rotation = rotation;
    this.scale = scale;
    this.layerDepth = layerDepth;
    Game1.EVENT_DRAW += this.Draw;

}
/// <summary>
/// Constructor for the map
/// </summary>
/// <param name="tex">the texture of the map</param>
/// <param name="scale">used to change the size of the drawing</param>
```

```
public Drawing(Texture2D tex, Vector2 scale)
 {
    this.tex = tex;
    this.scale = scale;
    this.position = Vector2.Zero;
    this.sourceRec = null;
    this.color = Color.White;
    this.rotation = 0;
    this.origin = Vector2.Zero;
    this.effect = SpriteEffects.None;
    this.layerDepth = 0;
    Game1.EVENT_DRAW += this.Draw;
 }
 #endregion

 #region Draw
 public void Draw() //a fucntion used for drawing on the screen
 {
    Stat.sb.Draw(tex, position, sourceRec, color, rotation, origin, scale, effect,
layerDepth);
 }
 #endregion

 #region makeTransparentColor
 /// <summary>
 /// a function used to change a specific color in a texture to be transparent
 /// </summary>
 /// <param name="color">The color to make transparent</param>
```

```
public void makeTransparentColor(Color color)

        {
            Color[] colors = new Color[tex.Width * tex.Height];
            tex.GetData<Color>(colors);
            for(int i = 0; i < colors.Length; i++)  if (colors[i] == color)
            colors[i] = Color.Transparent;
            tex.SetData<Color>(colors);
        }
        #endregion
    }
}
```

מחלקת Background:

מחלקה שיורשת מ-Drawing, תפקידה לעבור על הטקסטורה ולקבוע איזה סוג של קרקע כל מקום בטקסטורה על פי צבע הפיקסלים. כאשר הפיקסל שחור, הקרקע נקבעת להיות מסוג Road. כאשר הפיקסל בצבע ירוק, הקרקע נקבעת להיות מסוג Grass, וכאשר הפיקסל מכל צבע אחר, סוג הקרקע נקבע כ-Edge, שמסמל את קצה המפה. בעזרת המנגנון הזה אפשר להבין מתי המכונית מגיעה לקצה של הכביש, ולעצור אותה מלצאת מגבולות המשחק, דבר שיאפשר למשחק מציאותי יותר ומאתגר יותר.

```
#region using
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
#endregion

namespace OrProject
{
    class Background: Drawing
    {
        #region data
        public enum BackgroundType
            //the types of ground in the background
        {
            Road,
            Grass,
            Edge
        }
```

```
private BackgroundType[,] background;
 public BackgroundType this[float x, float y]
 {
    get
    {
       x /= scale.X;
       y /= scale.Y;
       if(x<0 || x> background.GetLength(0) || y<0 || y>=
         background.GetLength(1))
       {
          return BackgroundType.Edge;
       }
       return background[(int)x, (int)y];
    }
 }
 #endregion

 /// <summary>
 /// A constructor for making a background
 /// </summary>
 /// <param name="tex">The texture of the background</param>
 #region Constructor

 public Background(Texture2D tex)
    :base(tex, Vector2.One)
 {
    background = new BackgroundType[tex.Width, tex.Height];
    Color[] Colors = new Color[tex.Width * tex.Height];
    tex.GetData<Color>(Colors);
    for (int w = 0;  w < tex.Width;  w++)
       //this function puts the default background type as edge, if the pixel is
       black it sets it to road and if green to grass
    {
```

```
            for (int h = 0; h < tex.Height; h++)
                {
                    background[w, h] = BackgroundType.Edge;
                    if(Colors[w + h * tex.Width] == Color.Black)
                    {
                        background[w, h] = BackgroundType.Road;
                    }
                    if(Colors[w + h * tex.Width] == Color.Green)
                    {
                        background[w, h] = BackgroundType.Grass;
                    }
                }

            }
        }
        #endregion
    }
}
```

מחלקת Engine:

מחלקה שבעזרתה ניתן לבנות אובייקט של מנוע, עם הפרמטרים של המהירות העכשווית, מהירות ההאצה, ומהירות מקסימלית. בנוסף, יש במחלקה פעולה שבודקת האם הרכב נוסע קדימה או אחורה, אם הרכב נוסע קדימה, הפעולה מוסיפה למהירות העכשווית את מהירות ההאצה, וגם בודקת שהמהירות לא עברה את המהירות המקסימלית. אם הרכב נוסע אחורה הפעולה מורידה מהמהירות העכשווית את מהירות ההאצה, וגם בודקת שהמהירות העכשווית לא עברה את המהירות המקסימלית.

```
#region using
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
#endregion

namespace OrProject
{
    class Engine
    {
        #region data
        public float MaximumSpeed { get; set; }
        public float Speed { get; set; }
        public float AccelerationSpeed { get; set; }
        #endregion
```

```
/// <summary>
/// A constructor used to make an engine for the vehicle
/// </summary>
/// <param name="MaximumSpeed">Represents the maximum speed possible
    for the vehicle to reach</param>
/// <param name="AccelerationSpeed">Represents the rate the vehicle can
   accelerate</param>

#region Constructor

public Engine(float MaximumSpeed, float AccelerationSpeed)
{
   this.MaximumSpeed = MaximumSpeed;
   Speed = 0;
   this.AccelerationSpeed = AccelerationSpeed;
}
#endregion

#region Accelerate
/// <summary>
/// a function used to determine if the car is accelerating
/// </summary>
/// <param name="forward">used to determine if the vehicle is moving forward
   or backwards</param>
```

```
    public void Accelerate(Boolean forward)
    {
        if (forward)
        {
            Speed += AccelerationSpeed;
            if (Speed > MaximumSpeed) Speed = MaximumSpeed;
        }
        else
        {
            Speed -= AccelerationSpeed;
            if (Speed < -MaximumSpeed) Speed = -MaximumSpeed;
        }
    }
    #endregion
  }
}
```

מחלקת Vehicle:

מחלקה שיורשת מ-Drawing שבעזרתה ניתן לבנות אובייקט של כלי רכב.
לכלי הרכב יש משתנה של מנוע, של מהירות הרכב, של זווית הסיבוב, וכפתורים שמשמשים לתזוזת הרכב. בנוסף, יש פעולת Update שמאפשרת שינוי של זווית המכונית, בתנאי שהמכונית לא במקום. הפעולה גם קובעת האם הרכב נוסע אחורה או קדימה, ומשנה את המיקום שלו בהתאם למהירות ולזווית הסיבוב.

```
#region using
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
#endregion

namespace OrProject
{

    class Vehicle:Drawing
    {
        #region data
        public float deltaRotation;
        public Vector2 Velocity { get; private set; }
        public Engine engine { get; set; }
        public BasicKeys basicKeys { get; set; }
        #endregion
```

```
#region Constructor
/// <summary>
/// A constructor used for creating a vehicle
/// </summary>
/// <param name="tex">The texture of the vehicle</param>
/// <param name="position">The position you want to draw the vehicle
    in</param>
/// <param name="Origin">The point on the texture to refer to</param>
/// <param name="engine">The engine of the vehicle</param>
/// <param name="events">Used to know if you want to include the vehicle in
    the update process in Game1 class.</param>

public Vehicle(Texture2D tex, Vector2 position, Vector2 Origin, Engine engine,
bool events)
   : base(tex, position, Color.White, 0.0f, new Vector2(1, 1), 0.0f)
{
   this.engine = engine;
   this.Velocity = Velocity;
   if (events) Game1.EVENT_UPDATE += this.Update;
}
#endregion

#region Update
public void Update(GameTime gameTime) //the update function for the vehicle
{
   deltaRotation = 0;
   if(basicKeys.Right() && Math.Abs(engine.Speed) > 0.3f)
   {
      deltaRotation += 0.03f;
   }
```

```
        if (basicKeys.Left() && Math.Abs(engine.Speed) > 0.3f)
        {
            deltaRotation -= 0.03f;
        }
        rotation += deltaRotation;
        Vector2 direction = Vector2.Transform(Vector2.UnitX,
        Matrix.CreateRotationZ(rotation));
        Velocity = engine.Speed * direction;
        if (!(Stat.background[position.X + Velocity.X, position.Y + Velocity.Y] ==
        Background.BackgroundType.Road))
        {
            Velocity *= -1f;
            engine.Speed = 0;
        }
        if (basicKeys.Up()) engine.Accelerate(true);
        if (basicKeys.Down()) engine.Accelerate(false);
        position += Velocity;


    }
    #endregion
  }
}
```

מחלקת Camera:

מחלקה שמאפשרת יצירת אובייקט של מצלמה, שיכולה לעקוב אחרי אובייקטים מסוגים שונים. המחלקה מזיזה את המצלמה בעזרת הפעולה UpdateMatri.

```
#region using
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
#endregion

namespace OrProject
{
    class Camera
    {
        #region data
        public Matrix matri { get; private set; }
        public FocusObject focus { get; private set; }
        public Vector2 zooming { get; private set; }
        public Vector2 Position { get; private set; }
        Viewport vp;
        #endregion

        #region Constructor
        /// <summary>
        /// Constructor for camera
        /// </summary>
```

```
        /// <param name="vp">used to gather information about the size of the
            window</param>
        /// <param name="focus">The object you want the camera to focus on</param>
        /// <param name="zooming">Used to zoom the camera in or out</param>

        public Camera(Viewport vp, FocusObject focus, Vector2 zooming)
        {
            this.focus = focus;
            this.zooming = zooming;
            this.vp = vp;
            Position = new Vector2(0,0);
        }
        #endregion

        #region UpdateMatri
        public void UpdateMatri()
            /* A function used to apply a Matrix that first moves the object to 0,0 then
              applies a rotation, then applies the scale, and then moves the camera so
              the object will be in the middle */
        {
            matri = Matrix.CreateTranslation(-Position.X, -Position.Y, 0) *
                Matrix.CreateRotationZ(-focus.rotation * 0.1f) *
                Matrix.CreateScale(this.zooming.X, this.zooming.Y, 1f) *
                Matrix.CreateTranslation(vp.Width/2, vp.Height/2, 0);
            Position = Vector2.Lerp(focus.position, Position, 0.5f);
        }
        #endregion
    }
}
```

מחלקת BasicKeys:

מחלקה אבסטרקטית, שמטרתה להוות בסיס לשליטה על הרכב גם לאנשים וגם לבוטים. שחקנים שולטים ברכב בעזרת לחיצה על כפתור במקלדת, דבר שבוט לא יכול לעשות, ולכן דרושים מימושים שונים לפעולות שקובעות האם להזיז את הרכב.

מחלקת PlayerKeys היא מחלקה שיורשת מ-BasicKeys וממשת את הפעולות לשימוש של שחקנים, בעזרת בדיקה האם נלחץ כפתור במקלדת.

```
#region using
using Microsoft.Xna.Framework.Input;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
#endregion

namespace OrProject
{
    #region BasicKeys
    abstract class BasicKeys
    {
        public abstract bool Up();
        public abstract bool Down();
        public abstract bool Right();
        public abstract bool Left();
    }
    #endregion
```

```
#region PlayerKeys
class PlayerKeys : BasicKeys
{
    private Keys up, down, right, left;
    #endregion

    #region Constructor
    /// <summary>
    /// A constructor used to determine the keys used
    /// </summary>
    /// <param name="up">Represents the up key, for the vehicle it is used to move
it forward</param>
    /// <param name="down">Represents the down key, for the vehicle it is used to
move it backwards</param>
    /// <param name="right">Represents the right key, for the vehicle it is used to
rotate to the right</param>
    /// <param name="left">Represents the left key, for the vehicle it is used to
rotate to the left</param>

    public PlayerKeys(Keys up, Keys down, Keys right, Keys left)
    {
        this.up = up;
        this.down = down;
        this.right = right;
        this.left = left;
    }
    #endregion
```

```
        #region up
        public override bool Up()
        {
            return Keyboard.GetState().IsKeyDown(up);
        }
        #endregion

        #region Down
        public override bool Down()
        {
            return Keyboard.GetState().IsKeyDown(down);
        }
        #endregion

        #region right
        public override bool Right()
        {
            return Keyboard.GetState().IsKeyDown(right);
        }
        #endregion

        #region left
        public override bool Left()
        {
            return Keyboard.GetState().IsKeyDown(left);
        }
        #endregion
    }
}
```

מחלקת Stat:

מחלקה סטטית שתפקידה העיקרי הוא לאפשר ציור על גבי המסך בעזרת מחלקות שונות, שאינן מחלקת Game1. בנוסף, המחלקה יוצרת משתנה סטטי של רקע, על מנת להשתמש בו לרקע המסך, בלי הצורך ליצור אותו.

```
#region using
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Content;
using System.IO;
#endregion

namespace OrProject
{
    public delegate void DlgtUpdate(GameTime gameTime);
    public delegate void DlgtDraw();
    static class Stat
    {
        #region data
        public static ContentManager cm;
        public static SpriteBatch sb;
        public static Background background;
        #endregion
```

```
        #region Constructor
        /// <summary>
        /// a function used to bring the drawing capabilities from Game1 class to other
            classes
        /// </summary>
        /// <param name="cm">Used in order to be able to get data from the content
             folder</param>
        /// <param name="sb"></param>
        /// <param name="gd">The Graphics device is used in order to draw on
            screen</param>

        public static void Initialize(ContentManager cm, SpriteBatch sb, GraphicsDevice
gd)
        {
            Stat.cm = cm;
            Stat.sb = sb;
        }
        #endregion
    }
}
```