

שנקר-בי"ס גבוה להנדסה ולעיצוב

**הפקולטה להנדסה**

**המחלקה להנדסת תוכנה**

**TeamUP - CubeTris**

**פרוייקט גמר**

**מאת**

**ולדימיר לנדו וירוסלב סבירסקי**

**מנחה אקדמי: חנן גזין**

**מוגש כחלק מהדרישות לקבלת תואר ראשון**

**בוגר במדעים .(B.Sc.)**

22/06/2013

תוכן עניינים

[הכרת תודה 5](#_Toc367134811)

[תקציר מנהלים 6](#_Toc367134812)

[עברית 6](#_Toc367134813)

[אנגלית 6](#_Toc367134814)

[רשימת טבלאות וגרפים 7](#_Toc367134815)

[איור 1.0 – דיאגרת מצבים של ה Client Side: 7](#_Toc367134816)

[איור 2.0 – דיאגרמת תקשורת Server-Client: 8](#_Toc367134817)

[איור 2.1 – דיאגרמת מצבים של ה Server: 8](#_Toc367134818)

[איור 3.0 – Performance analysis 9](#_Toc367134819)

[איור 5.0 – Class Diagram without external libraries 11](#_Toc367134820)

[11](#_Toc367134821)

[מילון מונחים 12](#_Toc367134822)

[המטרות 13](#_Toc367134823)

[סיבות שהביאו לפרוייקט 13](#_Toc367134824)

[מטרות הפרוייקט 13](#_Toc367134825)

[מטרות לימודיות 13](#_Toc367134826)

[אפיון 14](#_Toc367134827)

[מטרת הפרויקט 14](#_Toc367134828)

[מטרת המשחק 14](#_Toc367134829)

[מצבי המשחק 14](#_Toc367134830)

[שליטה במשחק 15](#_Toc367134831)

[עריכת שלבים 16](#_Toc367134832)

[מסך כניסה 16](#_Toc367134833)

[מסך התחברות למשחק: 17](#_Toc367134834)

[מסך המשחק: 18](#_Toc367134835)

[מסך תוצאות: 19](#_Toc367134836)

[תיכון 20](#_Toc367134837)

[מבנה קבצי מערכת 20](#_Toc367134838)

[היררכיית קבצי קוד 21](#_Toc367134839)

[מבנה XML 22](#_Toc367134840)

[מבנה סצנה 24](#_Toc367134841)

[Screen Manager 25](#_Toc367134842)

[מבוא 26](#_Toc367134843)

[סקר ספרות\מערכות 27](#_Toc367134844)

[פיסיקה במשחק מולטיפלייר 27](#_Toc367134845)

[חישוב נקודות 28](#_Toc367134846)

[Documentation and coding standard 29](#_Toc367134847)

[שיטות ויישום המערכת 32](#_Toc367134848)

[UNITY 32](#_Toc367134849)

[C# 32](#_Toc367134850)

[Player.IO 32](#_Toc367134851)

[Command based server-client 32](#_Toc367134852)

[Optimization 32](#_Toc367134853)

[Resource pooling 32](#_Toc367134854)

[Alpha batching 33](#_Toc367134855)

[Performance driven design 33](#_Toc367134856)

[התאמת פיסיקה 33](#_Toc367134857)

[מערכת Events 33](#_Toc367134858)

[State machines 34](#_Toc367134859)

[MVC 34](#_Toc367134860)

[Matchmaking 34](#_Toc367134861)

[Custom gestures detection 35](#_Toc367134862)

[Object oriented design 35](#_Toc367134863)

[Source control 35](#_Toc367134864)

[בדיקות, תוצאות והערכה 37](#_Toc367134865)

[עמידה ביעדים שנקבעו 37](#_Toc367134866)

[לאפשר משחק Cross platform יחודי בין מכשירי מובייל, מחשב אישי וקונסולות 37](#_Toc367134867)

[רצון לייצר משחק חדש המשלב מספר זאנרים פופולריים מתחום משחקי המחשב 37](#_Toc367134868)

[התקמצעות בתחום הReal time client – server multiplayer 37](#_Toc367134869)

[להתנסות ביצירת Gestures חדשים עבור מצלמת הקינקט 38](#_Toc367134870)

[בניית פרוייקט בקנה מידה מקצועי בתחום משחקי המחשב 38](#_Toc367134871)

[Code complexity analysis 39](#_Toc367134872)

[בדיקות איכות תוכנה 39](#_Toc367134873)

[Performance analytics 39](#_Toc367134874)

[ניתוח סטטיסטי לבניית שלבים 40](#_Toc367134875)

[דיון: מסכנות ופיתוחים עתידיים 41](#_Toc367134876)

[תכנון נכון על מנת לעמוד ביעדים 41](#_Toc367134877)

[מולטיפלייר קבוצתי 41](#_Toc367134878)

[השמה למטרות חינוכיות 41](#_Toc367134879)

[התאמת המשחק לאנשים בעלי מוגבלויות 41](#_Toc367134880)

[מקורות 43](#_Toc367134881)

[נספחים 44](#_Toc367134882)

[תוכנה/דמו מוצר 45](#_Toc367134883)

# הכרת תודה

* תודה רבה לד"ר חנן גזית על התמיכה הרבה שלו בפרוייקט שלנו, ההכוונה המקצועית והעזרה בפיתוח הרעיון. ד"ר חנן גזית ליווה אותנו במהלך כל הפרוייקט, עזר לנו להתמקד בדברים החשובים ותרם מהידע הרב שלו בתחום

ה-Gamification על מנת לגרום למשחק להיות מהנה יותר.

* תודה רבה לד"ר אמנון דקל על התמיכה בראיון הפרוייקט ואישורו, המחלקה השתנתה מאד מאז הגעתו של אמנון ויש תחושה של הכרה בתחום משחקי המחשב כתחום הנדסי לצד התמחויות סטנדרטיות יותר בהנדסת תוכנה.
* אנו מכירים תודה לד"ר מרסלו שיכמן על הידע הרב שהוא לימד אותנו בתחום התכנות ותכנות מונחה עצמים. אנו מרגישים שבעזרתו הגענו לרמה המקצועית בה אנחנו נמצאים היום והבסיס שלמדנו ממנו מהווה תשתית לכל ההתפתחות המקצועית שלנו בתחום.
* תודה לד"ר איל שפר על הידע החשוב שהעביר לנו בקורס גראפיקה ממוחשבת, ידע זה משמש כבסיס לפיתוח מקצועי בתחום פיתוח משחקים.

# תקציר מנהלים

## עברית

המשחק CubeTris, הינו תוצר מיוחד ויחודי במינו. על מנת להבטיח זאת שלבנו סגנונות משחק נפוצים לתוך סגנון חדש המאפשר משחק רב-משתתפים מאתגר ומעניין. לקחנו את הבסיס המשחקי מסוג המשחקים- "Tower Defense ", מטרתו של השחקן במשחקים אלו היא להגן על הבסיס מול האויבים. את הרעיון המשחקי שלבנו עם הטטריס הישן והמוכר, בנוסף הוספנו אלמנטים משחקיים מז'אנר משחקי הפאזל על מנת לאתגר את השחקנים. התוצאה: בתחילת המשחק, השחקן משחק בסביבה הדומה לטטריס, אך לעומת הטטריס המקורי, המטרה של השחקן היא אינה לסדר את הלבנים בשורה שלמה מבלי להחסיר אף משבצת כמו במשחק המקורי, אלה להגיע לצורה מסוימת הנתונה לשחקן, על מנת לקבל את מלו הנקודות, ולאחר מכן להגן על המגדל אשר בנה.

כדי להבטיח כי התוצר הסופי הוא יחודי וחדשני בשוק המשחקים, המשחק נבנה כך שיעבוד על כל הפלטפורמות הנפוצות היום בעולם המשחקים. המשחק עובד בכ-90% מהטלפונים החכמים והטאבלטים שיש כיום בשוק, כמו כן ניתן להפעיל את המשחק גם בעזרת Microsoft Kinect.

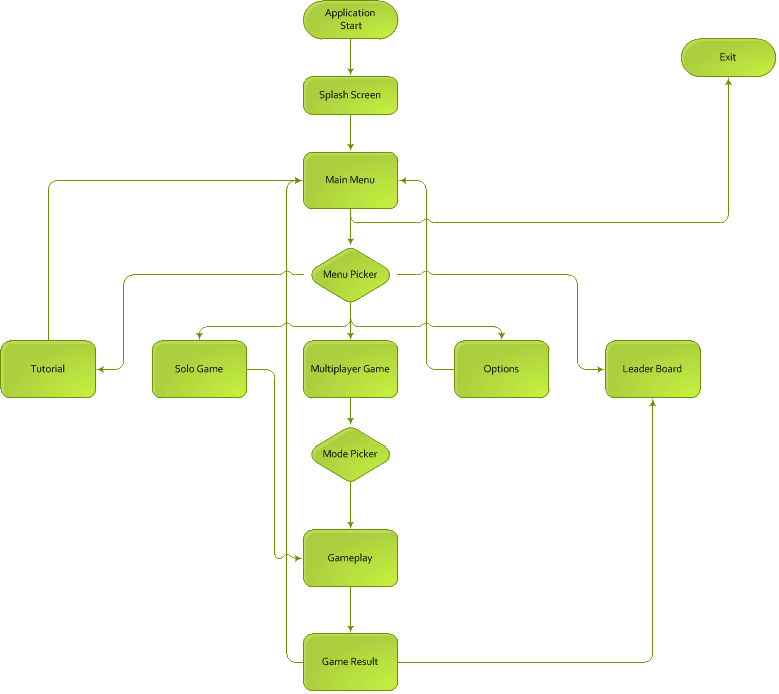
## אנגלית

CubeTris Game is a special and unique product. In order to ensure that uniqueness, we combined multiple popular game genres into a new interesting and challenging multiplayer game. We took Tower Defense where the player defends his base from multiple waves of enemies, and combined it with a good, old Tetris with an addition of puzzle concept to challenge the players. At the beginning, the player plays a Tetris style game, but the goal is not to remove lines, but to fulfill the game goal objectives, the next stage is when rockets are added into scene, and the player needs not only to build, but also to secure the built tower.

To ensure that our game is not only challenging but also one of a kind, we made it to fit as many platforms as possible, so the game can be deployed into 90% of smartphones and tablets, the game runs on most PC's, Mac's and mobile devices and also offering a unique experience to players which owns a Kinect camera.

# רשימת טבלאות וגרפים

## **איור 1.0 – דיאגרת מצבים של ה Client Side:**

****

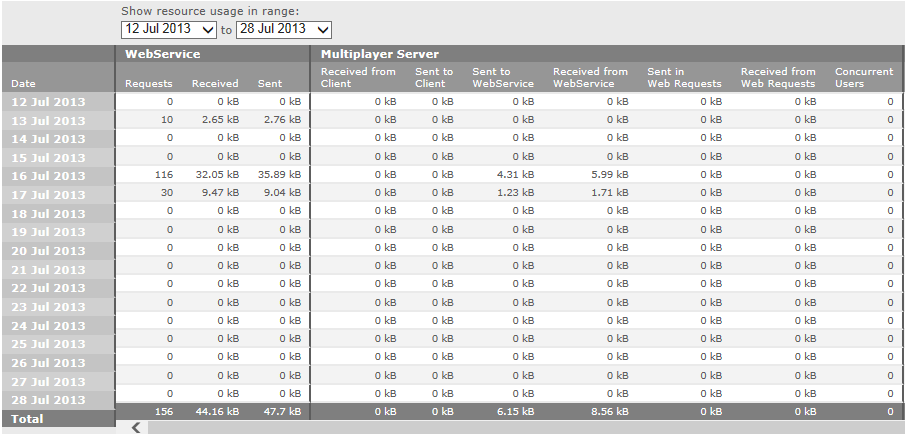
## **איור 2.0 – דיאגרמת תקשורת Server-Client**:



## איור 2.1 – דיאגרמת מצבים של ה Server:



## D:\Dropbox\Share_Slava (1)\Final Proj\profiler.PNGאיור 3.0 – Performance analysis

**איור 4.0 – Multiplayer traffic usage table to monitor server costs**

## איור 5.0 – Class Diagram without external libraries

## D:\Dropbox\Share_Slava (1)\Final Proj\cd.png

# מילון מונחים

Gamification - מנגנונים ורעיונות מעולם המשחקים שניתן לשלבם בתחומים שאינם משחקים.

Tetris – משחק קלאסי בו על השחקן לסדר בצורה מירבית בלוקים הנופלים רנדומלית.

Tower Defence - זאנ'ר פופולרי בתחום המשחקים הדיגיטליים בו על השחקן להגן על נכסיו מפני תוקפים.

iOS - פלטפורמת מובייל מבית Apple עליה רץ המשחק.

Android – פלטפורמת מובייל מבית Google עליה רץ המשחק.

Kinect – מצלמת זיהוי תנועה מבית Microsoft, המשחק מאפשר שליטה באמצעות מצלמה זו.

Builder - תפקיד במשחק, שחקן המשחק תפקיד זה אחראי בניית מגדל כפי שמופיע בהוראות השלב.

Defender – תפקיד במשחק, שחקן המשחק תפקיד זה אחראי על הגנת המבנה מפני טילים ואויבים.

Unity – מנוע משחק תלת מימדי בעל מספר exporters המאפשרים להוציא משחקים לפלטפורמות שונות, מבוסס על .NET ותומך בשפות תכנות הבאות: C#, JS, BOO.

Player.IO – ספריית עזר לכתיבת RealTime multiplayer server, מספקת רכיבים בסיסיים כגון sockets connection, יצירת חדרים ובסיס נתונים לשמירת נתוני משחק.

Physix – מנוע פיזיקלי שפותח על ידי Nvidia, המנוע נמצא בשימוש במשחקים רבים, הפיסיקה של Unity מבוססת על מנוע פיסיקלי זה ולכן גם הפיסיקה במשחק מבוססת עליו.

Spawner – רכיב במשחק בעל יכולת לבנות אובייקטים ולהוציאם לעולם המשחק.

# המטרות

## סיבות שהביאו לפרוייקט

* חוסר גיוון וחדשנות במשחקים דיגיטליים הקיימים והיוצאים לשוק בשנים האחרונות. אומנם כמות הכותרים החדשים הינה גדולה מאוד, אך רובם אינם מחדשים, אלה חוזרים על אותם אלמנטים משחקיים שכבר הוכיחו עצמם בעבר כרווחים.
* אי יכולת לשחק באותו משחק על פלטפורמות שונות. אני עדים לתופעה בה משחקים רבים ובינהם משחקי מולטי פלייר מוגבלים לאקוסיסטם ספציפי, (לדוגמא IOS) לדעתנו גישה זו מוטעת, ומגבילה את האינטראקציה והאפשריות החברתיות של השחקנים.

## מטרות הפרוייקט

* לאפשר משחק Cross platform יחודי בין מכשירי מובייל, מחשב אישי וקונסולות. המאפשר לשחקן חופש מוחלט בבחירת הפלטפורמה המועדפת עליו, ולהציע חוויה יחודית במשחק (טאץ', מקלדת, תנועה).
* רצון לייצר משחק חדש המשלב מספר ז'אנרים פופולריים מתחום משחקי המחשב על ידי כך להראות שאין סוף לדמיון בתחום משחקי המחשב ואין חייבים לדבוק בנוסחה קיימת על מנת לתת חוויה מהנה לשחקן.

## מטרות לימודיות

* התקמצעות בתחום הReal time client – server multiplayer . כחלק מהתואר בהנדסת תוכנה למדנו על תחום התקשורת בין מחשבים, אך לדעתנו לא קיבלנו מספיק נסיון מעשי בתחום, לכן החלטנו לאפשר לעצמנו לקבל את הנסיון שבעינינו דרוש לנו כדי להתקדם בתעשייה.
* להתנסות ביצירת Gestures חדשים עבור מצלמת הקינקט.
* בניית פרוייקט בקנה מידה מקצועי בתחום משחקי המחשב ושימוש בטכנולוגיות המקובלות בתעשיה. רוב הפרוייקטים בהם עסקנו במהלך התואר היו פרוייקטים בקנה מידה מצומצם עד בינוני, ורובם אינם שיקפו כללו אינטראקציה בין מערכות תוכנה שונות, רוב הפרוייקט שצויינו לעיל כללו בעיקר התעסקות במערכת סגורה מסויימת ללא אינגרציות עם מערכות חיצוניות. לדוגמא: תוכנית המרת מטבע בJAVA.

# אפיון

## מטרת הפרויקט

לאפשר משחק Cross platform יחודי בין מכשירי מובייל, מחשב אישי וקונסולות. המאפשר לשחקן חופש מוחלט בבחירת הפלטפורמה המועדפת עליו, ולהציע חוויה יחודית במשחק (טאץ', מקלדת, תנועה).

בנוסף הרצון לייצר משחק חדש המשלב מספר ז'אנרים פופולריים מתחום משחקי המחשב על ידי כך להראות שאין סוף לדמיון בתחום משחקי המחשב ואין חייבים לדבוק בנוסחה קיימת על מנת לתת חוויה מהנה לשחקן.

## מטרת המשחק

על השחקן לבנות מגדל בהתאם למטרה המוצגת על המסך, למשימה זו מוקצב X זמן.

על מנת לבנות את המגדל המושלם, על השחקן לתפוס את קוביות הבניה אשר נופלות משמי המשחק (החלק העליון לשל המסך) תוך כדי נפילתן, ולגורר אותן למקומות הכי קרובים לצורת המטרה.

בנוסף לכך, על השחקן להגן על מגדלו מטילים שנופלים מהחלק הלעיון של המסך. מכיוון שהמשחק הינו משחק לילדים, אין בו אפשרות להפסיד ובכל מקרה השחקן מגיע למצב תוצאות ושם הוא מקבל סטטוס בהתאם לכמות הנקודות שצבר, כיוון שאין אפשרות להפסיד, כל שחקן מקבל הרגשה כי זכה. שימוש במנגנון גיימייקיישן אמור להבטיח הנאה לשחקנים מתחילים ואתגר לשחקנים מתקדמים, שימוש במנגון הגיימיפקיישן אשר צויין לעיל מבטיח הנאה צרופה לשחקנים המתחילים, ובנוסף לזאת מעורר תחושת סקרנות ורצון לגלות וריאציות שונות לסיום המשחק עבור המשתמשים המתקדמים (גילוי סטטוסים חדשים).

## מצבי המשחק

במשחק קיימים שני תפקידים:

* תפקיד ה-BUILDER : הבונה את המגדל מהצורות הנופלות מקצה העליון של המסך, בהתאם לצורה המוצגת על גבי מסך המשחק (צורת המטרה של השלב הנוכחי).
* תפקיד ה- DEFENDER : המגן על המגדל הנבנה על ידי ה-BUILDER מפני איומים הטלים השונים אשר מאיימים להרוס את המגדל, יחד עם זאת עליו לאסוף בונוסים מתגמלים.

במשחק לשחקן היחיד:

השחקן מבצע את שני התפקידים בעצמו, המשחק לשחקן היחיד לא מאפשר לשחקן להתקדם בשלבים ולעלות רמות מחשק, עקב הרצון שלנו כמפתחים לעודד שיתוף פעולה ולהניע שחקנים ממקומות שונים בעולם לאינטראקציה חברתית נעימה, לסיכום ניתן להגיד כי מצב המחשק לשחקן יחיד הינו נועד על מנת לתת לשחקן המתחיל "טעימה" ממה שממתין לו במשחק המולטי פלייר.

במשחק המולטי פלייר:

אחד השחקנים מבצעה את תפקיד ה-DEFENDER והשחקן השני את תפקיד ה-.BUILDER עליהם לשתף פעולה בצורה המירבית והאיכותית ביותר על מנת לעמוד איתנים מול האתגרים האינוספיים אותם מציב בפניהם המשחק.

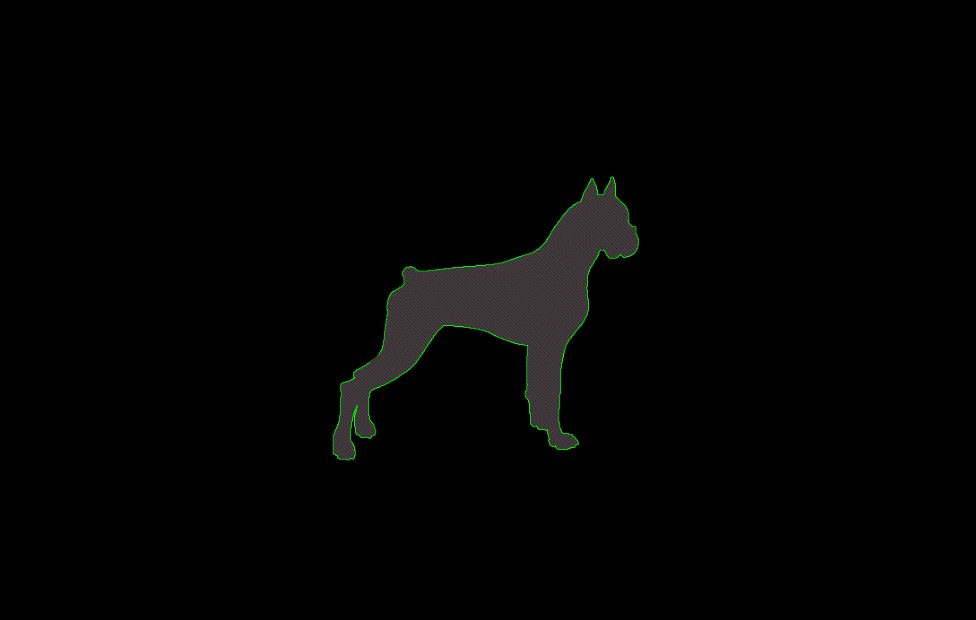
## שליטה במשחק

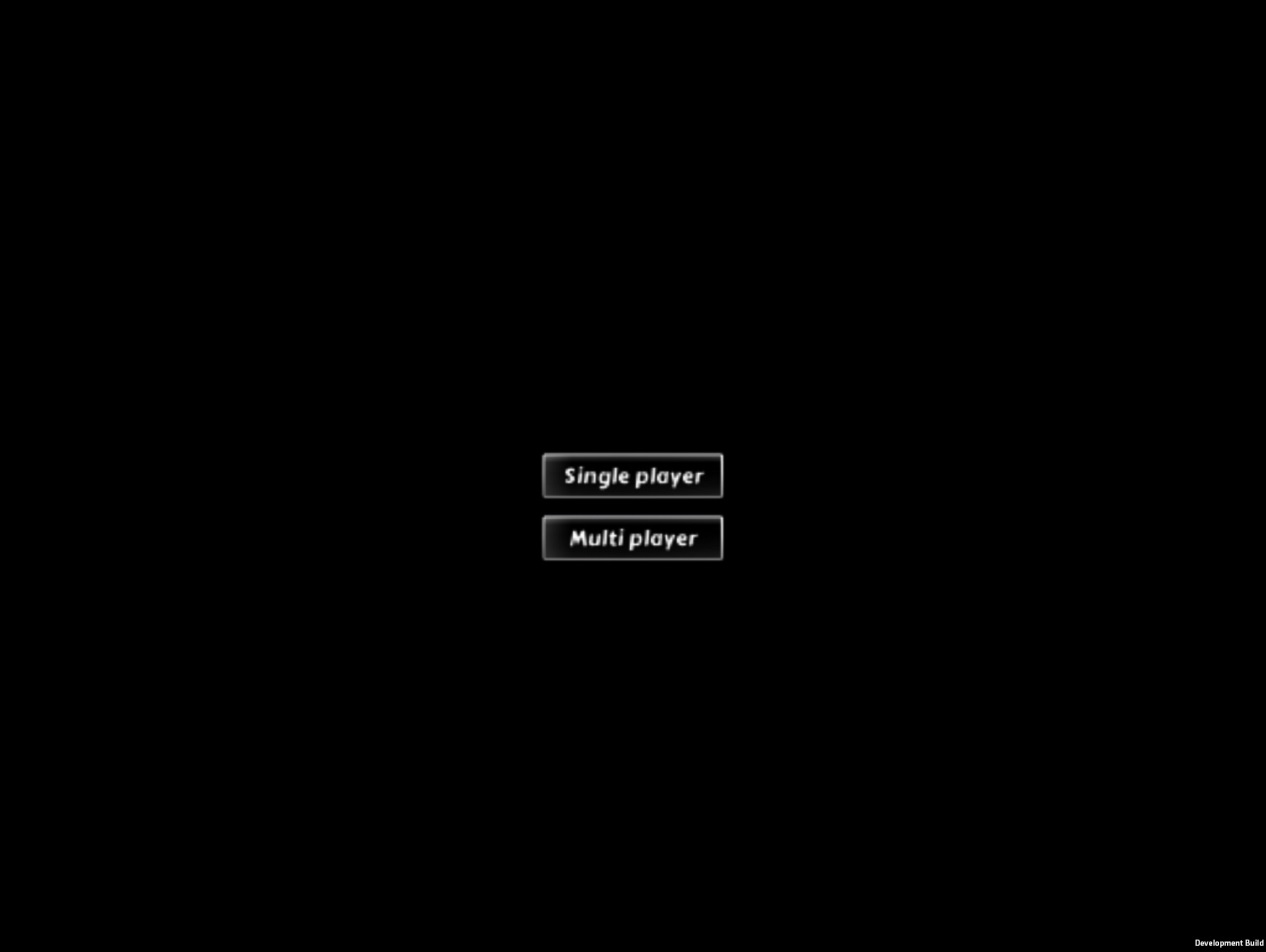
קיימים שלושה סוגים של שליטה במשחק:

* עכבר - כאשר מריצים את המשחק על גבי ה- PC או הMAC העמצאי בעזרתנו ניתן לשלוט במחשק הוא העכבר בעצמאותו ניתן לממש גם את תפקיד הבילדר וגם את תפקיד הדפנדר. כרגע השימוש במקלדת אינו נתמך, כמו כן כרגע לא נתמכים התקני קלט פלט נוסף, לדוגמא: ג'וייסטיק.
* מסך מגע - כאשר מריצים את המשחק על גבי מכשירים הניידים, סמארטפונים וטאבלטים על ידי שימוש במסך המגע ניתן לממש גם את תפקיד הבילדר וגם את תפקיד הדפנדר.
* קינקט - כאשר מריצים את המשחק על גבי ה-PC או הXBOX , (תאורתית, עדיין לא נבדק) ניתן להשתמש במצלמת הקינקט כעמצאי קלט פלט לשליטה במחשק. על ידי שימוש בקינקט ניתן לממש את תפקיד הדפנדר בלבד, (תפקיש הבלדר עדיין בפיתוח).

## עריכת שלבים

## מסך כניסה





במסך הכניסה מופיע תפריט המאפשר בחירה בין שני הסוגים של המשחק, Single Player או MultiPlayer. האפשרות הנבחרת תסומן בצע ירוק.

## מסך התחברות למשחק:



מסך המתנה לחיבור מופיע כאשר נוצר קשר עם השרת, והמארח ממתין לתשובה. במסך זה מופעלת אנימציה של טעינה (גלגל מסתובב), במידה במידה והתקשורת עם השרת נכשלת עקב סיבה כלשהי (טיים אאוט, שגיאה, בעיות חיבור וכו') השחקן יועבר למסך שגיאה אשר מפרט בפירוט את השגיאה שהתרחשה וממליץ על דרך מהירה לפתרון הבעיה. במידה והחיבור צלח השחקן מועבר למסך המשחק.

## מסך המשחק:



את מסך המשחק ניתן לחלק לשני חלקים עקריים:

* חלק המציג אינפורמציה:
  + ניקוד
  + מן משחק
  + FPS
* חלק המציג את המשחק עצמו וכולל את האלמנטים הבאים:
  + צורת מטרה- הצורה אותה על השחקנים לבנות על מנת לקבל ציון מירבי.
  + בלוקי בניה הנופלים מהחלק העליון של המסך.
  + טילים אשר נורים מהחלק העליון של המסך.
  + בונוסים- תוספות ופרסים אשר על השחקנים לאסוף.
  + זנב שחקן- אובייקט ויזואלי העוקב אחר תנועות השחקן על מנת לידע את השחקנים במיקום אחד של השני.

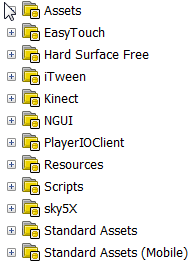
## מסך תוצאות:



בסיום של כל משחק, כל אחד מהשחקנים מגיע למסך התוצאות, במסך זה השחקן רואה את הניקוד אשר צבר, ואת ה-RANK שקיבל בהתאם לניקוד.

# תיכון

## מבנה קבצי מערכת

קבצים גרפיים של הפרוייקט

ספריית עזר לעבודה עם מסכי מגע

ספריית עזר של Shaders

ספריית Easing

ספריית עזר לעבודה עם Kinect

מערכת Gui

ספריית עזר לעבודה עם צד שרת

Prefabs וקבצי קונפוגורציה

רקעים

## D:\Dropbox\Share_Slava (1)\Final Proj\Screens\scripts.pngהיררכיית קבצי קוד

Connection screen implementation

Constant values for connection methods

Wrapper class for Player.IO connection API

Abstract interface to inherit from when implementing different control types

Manages currently selected controls interface

Handles Kinect input

Handles touch taps

Handles touch drag

Abstract figure to implement when creating game figure

Block figure class implementation

Unused

Abstract rocket to derive from when creating a rocket

Explosive rocket type

Non Explosive rocket type

Spawn types which are passed when spawner spawns a figure

Bonus spawn type

Figure spawn type

Rocket spawn time

General figure spawner which can handle every figure type

Unused

Game manager class which manages movement between stuff and general stuff

Global game properties

Script to make sure the every collision is detected, add an additional raycast test

Calculates game fps

Editor

General math equations

General helpers for Unity and C#

Helpers utils to work with XML files

Manages player achievements

Application level manages (pause, loose focus and etc events)

All game delegates and events are declared here

Adds localization to label at runtime

Manages localizable strings

Manages Music

Manages player

Resource pooling for prefabs, to save instantiations at runtime

Manages game sounds

Credits screen – not implemented

Game Over screen

Gameplay screen (AKA HUD)

Level selection screen – not implemented

Main menu screen

Options screen – not implemented

Remove ads screen – not implemented yet (monetization)

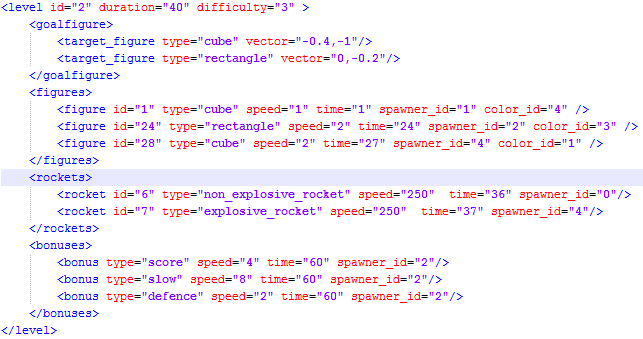
Room selection screen – hidden at the moment due to automatic matchmaking

Splash screen – shown when game launches

Stats screen – not implemented

MenuManager – manages transitions between menus, using stack data type to store navigation

## מבנה XML



מקבץ ההגדרות היחודיות עבור שלב מסויים יורדות מהשרת בצורה קובץ XML אשר מגדיר את כל ההתנהגות של השלב הנוכחי ונשלח לשחקן בעת סיום תהליך MATCHMAKING.

הקובץ מורכב מהאלמנטים הבאים:

* הגדרות כלליות של השלב
* הגדרות צורה סופית (המטרה אליה שואפים)
* הגדרת ה SPAWNING של הצורות
* הגדרת ה SPAWNING של הטילים
* הגדרת ה SPAWNING של הבונוסים

האלמנטים שצויינו לעיל מאפשרים לעורך המשחק להזין מאות שלבים חדשים בקלות וללא צורך ב Deploy! אזי תהליך העידכון שקוף מבחינת המשתמש ואינו מצריך ממנו הורדת גרסא חדשה של התכונה.

XML Structure:

level – parent node

id – level unique ID

duration – total duration of the level in seconds

difficulty – level difficulty, should match difficulties within player profile

goal\_figure – goal figures parent node

target\_figure – single node of parent figure

type – figure type: cube, rectangle, etc…

vector – position vector in screen coordinates

figures – game figures parent node

figure – represents a single game figure

id - unique ID per game

type - figure type: cube, rectangle, etc…

speed – launching speed

time – spawn time, should be smaller than level time

spawner\_id – spawner ID to spawn the figure within

color\_id – figure target color

rockets

rocket

id - unique ID per game

type - rocket type: explosive, non-explosive, atomic, etc…

speed – launching speed

time – spawn time, should be smaller than level time

spawner\_id – spawner ID to spawn the figure within

bonuses

bonus

id - unique ID per game

type – bonus type: +score, inc defence, slow objects

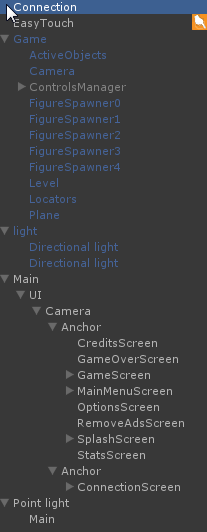
speed – launching speed

time – spawn time, should be smaller than level time

spawner\_id – spawner ID to spawn the bonus within

duration – bonus effect duration in seconds

## מבנה סצנה

Manages connection

Manages touch input

Main game object – stays the whole runtime

Parent to contain allocated objects

3D perspective camera for game scene

Control classes

Spawners

Level manager – changes upon each new level

Persistent locators

Floor plane

Light for occlusion

Menu manager

2D orthographic camera

Credits screen

Game over screen

Game screen (HUD)

Main menu screen

Options screen

RemoveAdsScreen – not implemented

StatsScreen

Connection message

## Screen Manager

הScreenManager אחראי על ניהול מעברים בין מסכים, שמירת מצבים של מסכים אחרונים לצרכי Navigation אחורי, אתחול וניקוי המסכים.

המודל לפי רכיב זה נבנה לפי עקרונות ה Navigation של מערכת ה iOS, להלן הנקודות החשבונות של רכיב זה:

* שמירת Navigation במבנה נתונים מסוג Stack
* אפשרות לחזור אחורה עד למסך הראשון ששמור
* ניתן לנקות את ה Navigation stack על מנת למנוע אפשרות חזרה
* כל הקריאות ל Navigation מבוצעות דרך ScreenManager.
* אפשרות להעברת מפה של ערכים מסוג < string, object > בקריאה לתצוגת מסך.
* אנומרציה של כל המסכים הקיימים על מנת למנוע קריאות למסכים לא קיימים

כל מסך במשחק יורש מהמחלקה BaseScreen המכילה מתודות אבסטרקטיות לשימוש המסך הספציפי, פתרון זה מאפשר שמירה גנרית של המסכים ללא צורך בידיעת המחלקה של המסך הספציפי, כל מסך יכול לממש את המתודות הבאות:

* Activate – נקראית כאשר המסך הופך לפעיל ומקבלת מפה של ערכי קלט
* Deactivate – נקראית כאשר המסך הופך ללא פעיל
* Init – אחראית על קישור רכיבים ויזואליים ללוגיקה

# מבוא

בפרוייקט זה אנו מנצלים את היכולות החדשניות של עולם פיתוח המשחקים המאפשרות לנו לפתח משחקים תלת-מימדים על בסיס מנוע גרפי קיים ויתרה על כך מאפשרות לעשות שימוש במנוע פיזי קיים עם מינימום התאמות.

עולם פיתוח המשחקים הינו עולם דינאמי ומודולרי ענק אשר משתנה במהירות רבה, כל מספר חודשים אנו עדים לטכנולוגיות מהפכניות שמשנות את תמונת המצב ודורשות מהמפתחים ללמוד שפות חדשות ולהתמקצע בכלי פיתוח חדשים.

בעבר נהגו חברות פיתוח המשחקים לבנות את מנועי המשחק מ 0 עבור כמעט כל פרוייקט, לשם כך נדרש כוח אדם רב. בנוסף מכיוון שכל מנוע היה יחודי במינו, נאלצו המפתחים לפתח כלים יחודיים למנוע ולבזבז זמן יקר על פיתוחים שלא ימשיכו לפרוייקטים הבאים.

עקב מגבלות הזמנים והכוח אדם, בחרנו לדגול בשיטה של לעבוד עם מנוע קיים ולהשתמש בplugins לדברים בסיסיים על מנת להתרכז בעיקרי הפרוייקט ולא בפיתוח הכלים הצדדיים:

* כלי DEBUG
* כלים לבדיקות עומסים
* EXPORTERS
* SCENE EDITOR

המטרה העיקרית הייתה לבנות מוצר אשר ישמח גם את הילדים וגם את המבוגרים, משהו שייתן הזדמנות להנות ממנו לגל העולמות, לאפשר לשחקנים ממקומות שונים לשחק יחד, בלי לעשות את החוויה למשעמם עבור אחד הצדדים, על מנת לעשות זאת נקטנו במספר צעדים:

* שימוש בסגנונות קיימים על מנת ליצור סגנון יחודי ומהנה.
* מערכת התאמת משחקים וניקוד שתבצע התאמות מירביות בין שחקנים.
* תכנון שליטה יחודית ושימוש ביחודיות של כל פלטפורמה.
* פנייה לקהל שחקנים רב ככל שניתן ממיקומים גאוגרפיים שונים ועל פלטפורמות משחק שונות.

אחרי נסיונות רבים, בחרנו להשתמש ביוניטי, בשביל המנוע הגרפי והפיסיקלי, יוניטי זאת פלטפורמת פיתוח המאפשרת פיתוח משחקים על בסיס מנוע קיים במהירות ונוחות. הבחירה הייתה נכונה, ואפשרה לנו לעשות דיפלוי לפלטפורמות רבות. הבחירה שלנו ביוניטי בנוסף אפשרה לנו קנייה של אסטים מוכנים עבור המודלים התלת מימדיים ללא צורך באומן תלת מימד.

על מנת ליצור חוויה אולטימטיבית נאלצנו לפתח צד שרת יחודי בעצמנו (לא מצאנו מוצר קיים שענה על צרכינו) הנותן את חווית המולטיפלייר, מערכת ניקוד אשר מבצעת התאמה בין שחקנים על בסיס ניקוד ופרמטרים נוספים כגון ותק, גיל, מין וכו'. בסיום כל שלב השחקנים מקבלים נקודות בהתאם להצלחתם במשחק, ובהתאם לנקודות השחקנים מקבלים בונוסים ודרגות, בכניסה עתידית, שחקן שקבל בעבר דרגה מסויימת יצוות על ידי התוכנה לשחקן בעל דרגה דומה, בהתאם דרגת הקושי של השלב תותאם לשני השחקנים על ידי התוכנה, זאת על מנת להאתים את דרגת הקושי לשחקנים, ולמנוע ממצב בו המשחק קשה מדי, ועלול לגרום לכשלונות ובהמשך לנטישה. ובהתאם גם כדי למנוע מצב בו המשחק קל מדי עבור שחקן מתקדם, מה שיצור שעמום וחוסר עניין במשחק.

ניתן להריץ את המשחק על גבי דיבייסים שונים, סמרטפונים, טבלטים, מחשבים ביתיים, לפטופים ומערכות משולבות קינקט. כמובן שאנחנו גם מאפשרים משחק מולטיפלייר בין הדיבייסים השונים, כך ששחקו אחד יכול לשחק על טאבלט בזמן שחבר שלו משחק עח גבי מחשב ביתי וכדומה. בגלל שיש הבדל בקושי המשחק, יש שינויים בכל גרסה לפי הפלטפורמה, כך שבמסך הקטן של טלפון אזור התפיסה של בלוק הוא גדול(בגודל יחסי) לעומת איזור התפיסה של המחשב הביתי.

במהלך הפיתוח של הפרוייקט נעמדנו בפני דילמה על מערכת לבניית שלבים דינאמית שמצד אחד צריכה להיות נוחה ל Game designer האחראי על בניית השלבים, מצד שני לא רצינו שמערכת כזו תתפוס הרבה מקום על כונן קשיח כדי לא להעלות את גודל המשחק ולקצר זמני טעינה.

הפתרון הכי יעיל שמצאנו הוא טעינת מידע מקבצי XML בפורמט נוח לעורך המשחק, פתרון זה מאפשר זמני טעינה מינימליים ביחד עם רמת שליטה מירבית עבור העורך.

# סקר ספרות\מערכות

## פיסיקה במשחק מולטיפלייר

פיסיקה הינה גורם חשוב בתורת משחקי המחשב ומוסיפה לריאליסטיות של המשחק ולשיפור האינטרקציה ותחושת השליטה של השחקן.

ישנן שתי שיטות מקובלות בתעשייה למימוש מנוע פיסיקלי במשחקי מולטיפלייר:

* המנוע הפיסיקלי רץ על השרת ומעדכן את כל השחקנים כל X זמן במצב הפיסיקלי של כל הסצנה, בשימוש בשיטה זו, כל שחקן מידע את השרת עם הקלט שלו והשרת אחראי על פענוח כל הקלטים ו עדכון המצב הפיסיקלי.

יתרונות של שיטה זו:

* + סנכרוניות מלאה בין השחקנים והשרת – כל שחקן יודע מצב אמיתי של מה שקורה סביבו.
  + אין תלות בין גורמים של מחשבי שחקנים לבין המנוע הפיסיקלי.

חסרונות:

* גודל המידע המועבר בכל פריים גדול מאד מכיוון שהוא מכיל את המצב הכולל של המערכת.
* פספוס מספר פריימים יגרום ליציאת שחקן מסנכרון.
* עומס חישובי רב על השרת.
* כל שחקן מריץ מנוע פיסיקלי בעצמו ואין קשר פיסיקלי ישיר בין השחקנים, בשיטה זו כל שחקן מדווח על מיקומו לשרת והשרת רק אחראי להעביר את מצבם הנוכחי של השחקנים ואובייקטים בעולם המשחק ללא התחשבות בפיסיקה.

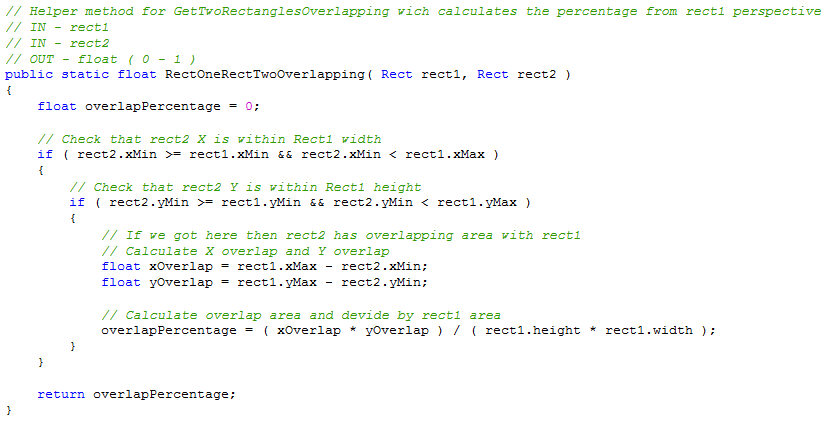
יתרונות של שיטה זו:

* + גודל הפתיל הרץ על השרת קטן יחסית ולא יוצר עומס.
  + כמות המידע המועברת ברשת בכל פריים קטנה משמעותית ביחס לשיטה הקודמת.
  + כאשר שחקן מפספס מספר עדכונים מהשרת, הוא עדיין ידע לזוז בצורה פיסיקלית תקינה בזכות המנוע הפיסיקלי הלוקאלי.
  + כמות החישובים והלוגיקה אשר מתבצעת בצד שרת, קטנה בהרבה יחסית לשיטה הקודמת, השרת משמש רק לצורכי ניתוב וכמעט לא מתבצעים בו חישובים.

חסרונות:

* יכול להיווצר חוסר סנכרון בין השחקנים בגלל אי דטרמניסטיות של המנועים הפיסיקיליים ותלותם בגורמים רבים כגון: מעבד, זמן מקומים של השחקן, LATENCY וגורמים נוספים...
* ייצוג שגוי של המתרחש בסצנת המשחק, לדוגמא: שחקן א הפעיל אבן הנופלת מבניין, שחקן ב' איבד לא מסונכרן על נפילת האבן ולכן המנוע הפיסיקלי שלו לא יחשב את הפגיעה של האבן ולכן לא יופעל TRIGGER מסויים נקרא לאחר נפילת האבן.

## חישוב נקודות



פונקציה המחשבת אחוז חיתוך בין 0 ל 1 של מלבן מסויים ביחס למלבן אחר.

מכיוון שהמשחק הינו תלת מימדי, אנו עושים הטלה למרחב מסך ממרחב תלת מימדי ואז משווים מחפשים את אחוז החיתוך בין הצורה שהשחקן בנה לצורה הרצויה, לבסוף עושים ממוצע חיתוך לכל המלבנים ולפיו נקבע ציון השחקן.

## Documentation and coding standard

במהלך הפרוייקט הקפדנו על הסטדנרטים של שפת c# כפי שמוגדרים ב MSDN על ידי Microsoft:

[Naming Conventions](javascript:void(0)):

var currentPerformanceCounterCategory = new System.Diagnostics.

PerformanceCounterCategory();

You do not have to change the names of objects that were created by using the Visual Studio designer tools to make them fit other guidelines.

[Layout Conventions](javascript:void(0)):

* Use the default Code Editor settings (smart indenting, four-character indents, tabs saved as spaces.
* Write only one statement per line.
* Write only one declaration per line.
* If continuation lines are not indented automatically, indent them one tab stop (four spaces).
* Add at least one blank line between method definitions and property definitions.
* Use parentheses to make clauses in an expression apparent, as shown in the following code.

if ((val1 > val2) && (val1 > val3))

{

// Take appropriate action.

}

[Commenting Conventions](javascript:void(0)):

* Place the comment on a separate line, not at the end of a line of code.
* Begin comment text with an uppercase letter.
* End comment text with a period.
* Insert one space between the comment delimiter (//) and the comment text, as shown in the following example.

// The following declaration creates a query. It does not run

// the query.

* Do not create formatted blocks of asterisks around comments.

[Implicitly Typed Local Variables](javascript:void(0)):

Do not use [var](http://msdn.microsoft.com/en-us/library/vstudio/bb383973.aspx) when the type is not apparent from the right side of the assignment.

// When the type of a variable is not clear from the context, use an

// explicit type.

int var4 = ExampleClass.ResultSoFar();

Do not rely on the variable name to specify the type of the variable. It might not be correct.

// Naming the following variable inputInt is misleading.

// It is a string.

var inputInt = Console.ReadLine();

Console.WriteLine(inputInt);

Avoid the use of var in place of [dynamic](http://msdn.microsoft.com/en-us/library/vstudio/dd264741.aspx).

Use implicit typing to determine the type of the loop variable in [for](http://msdn.microsoft.com/en-us/library/vstudio/ch45axte.aspx) and [foreach](http://msdn.microsoft.com/en-us/library/vstudio/ttw7t8t6.aspx) loops.

The following example uses implicit typing in a for statement.

var syllable = "ha";

var laugh = "";

for (var i = 0; i < 10; i++)

{

laugh += syllable;

Console.WriteLine(laugh);

}

[Arrays](javascript:void(0)):

Use the concise syntax when you initialize arrays on the declaration line.

// Preferred syntax. Note that you cannot use var here instead of string[].

string[] vowels1 = { "a", "e", "i", "o", "u" };

// If you use explicit instantiation, you can use var.

var vowels2 = new string[] { "a", "e", "i", "o", "u" };

// If you specify an array size, you must initialize the elements one at a time.

var vowels3 = new string[5];

vowels3[0] = "a";

vowels3[1] = "e";

// And so on.

[Delegates](javascript:void(0)):

Use the concise syntax to create instances of a delegate type.

// First, in class Program, define the delegate type and a method that

// has a matching signature.

// Define the type.

public delegate void Del(string message);

// Define a method that has a matching signature.

public static void DelMethod(string str)

{

Console.WriteLine("DelMethod argument: {0}", str);

}

// In the Main method, create an instance of Del.

// Preferred: Create an instance of Del by using condensed syntax.

Del exampleDel2 = DelMethod;

// The following declaration uses the full syntax.

Del exampleDel1 = new Del(DelMethod);

[Event Handling](javascript:void(0))

// Using a lambda expression shortens the following traditional definition.

public Form1()

{

this.Click += new EventHandler(Form1\_Click);

}

void Form1\_Click(object sender, EventArgs e)

{

MessageBox.Show(((MouseEventArgs)e).Location.ToString());

}

# שיטות ויישום המערכת

## UNITY

בחרנו ב Unity3D כמנוע לבניית המשחק מכיוון שהוא מציע סביבת עורך עשירה, מאפשר לבצע Export למספר פלטפורמות רב: iOS, Android, PC, Mac, etc…

בנוסף בגלל הותק והפופולריות של המנוע, ישנן ספריות עזר רבות שמקצרות את העבודה על המשחק.

## C#

בחרנו ב C# כשפה המועדפת בגלל נוחות הכתיבה, נסיון אישי, תמיכה טובה ב OOP מכיוון שהבסיס של השפה הוא OOP לעומת שפות כדוגמת JS, ותמיכה ב delegates.

## Player.IO

השתמשנו בספריה זו לצורך טיפול low level בתקשורת עם השרת, הספרייה מספק פונקיציאות עזר לשליחה\קבלת בקשות עם טיפוסים נפוצים, פתיחת חדרים וכו...

## Command based server-client

צד השרת במשחק מבוסס על פקודות כאשר כל פקודה היא מחרוזת יחודים שמוגדרת בserver ו ב client.

בכל בקשה או תשובה מהשרת מוסיפים את מחרוזת הפקודה הנוכחית למילון הבקשה, מה שמאפשר קביעה ודאית באיזו פקודה מדובר ללא צורך לזכור מצבים קודמים.

בכל מצב משחק יש פקודה או מספר פקודות שה client מחכה לקבל ואלה הפקודות היחודות איתם ידע להתמודד במצב נתון.

כאשר מבוצעת פעולה כלשהי שדורשת תשובה מהשרת בצד client, מתבצעת הרשמה ל event של תשובה גנרי שמוגדר במחלקה Connection.

## Optimization

### Resource pooling

אחת הבעיות הנפוצות ביותר בפיתוח משחקים היא הקצאת משאבים בזמן ריצה, ב Unity תהליך זה נקרא Instantiation, בתהליך זה מתבצעת קריאה מהדיסק שהיא פעולה יקרה בזמן אמת ונוצר אובייקט לפי הגדרתו בדיסק, לדוגמא קוביה תלת מימדית.

פעולות אלה כמעט ולא מורגשות כאשר מדובר באובייקטים בודדים אך כאשר יש צורך במאת ואלפי אובייקטים, פעולה זו תעצור את רצף המשחק לזמן ממושך ותפגע בחווית המשתמש.

על מנת למנוע תופעה זו השתמשנו במנגנון הנקרא Resource pooling, הטוען את האובייקטים לזכרון בזמן טעינה ראשונית והופך אותם ללא פעילים, כאשר המשחק מעוניין להשתמש באובייקט מסוג מסויים, הוא ניגש למאגר האובייקטים ומושך אותו מהמאגר. לאחר סיום השימוש באובייקט הוא אינו מושמד אלה חוזר למאגר. כאשר הדרישה לאובייקטים מסוג מסויים עולה על גודל המאגר, יוצרים העתק של פרוייקט במאגר ולא מבצעים קריאה מהדיסק.

### Alpha batching

בעיה נוספת שבה נתקלנו שפגעה בביצועי המשחק, היא שאובייקטים בעלי shaders התומכים בטקסטורה עם alpha ששונה מ 1, לא עשו batching, זאת אומרת לא רונדרו בקריאה אחת אלה כל אחד מהם רונדר בקריאה נוספת, למרות היותם העתקים של אותו אובייקט.

לאחר חקירה מרובה באינטרנט הגענו למסקנה שמדובר בבעיית Z order ב Unity, המנוע לא ידע לרנדר מספר אובייקטים עם Z שונה באותו draw call.

על מנת לפתור זאת כתבנו shaders חדשים בשביל אובייקטים בעלי טקסטורות שקופות והגדרנו בהם סדר רינדור קבוע וכך כל האובייקטים שמשתמשים באותו shader, ירונדרו באותו Z pass ומנגנון ה batching יוכל לזהות את היותם העתקים של אותו אובייקט מקור ולרנדר אותם ב draw call אחד.

## Performance driven design

בתחילת הפרוייקט חששנו מאד מבעיות ביצועים על מכשירי מובייל, לכן לאחר כל מודול שפיתחנו, דאגנו לבדוק את המשחק על ידי הרצת profiler על מכשיר Android לא חזק במיוחד ולוודא שהביצועים אינם יורדים מ 30 FPS.

## התאמת פיסיקה

למראות שמנוע פיזיקלי של יוניטי ענה על רוב הדרישות שלנו, אך היו דברים שגרמו לבעיות. בגלל החוסר דטרמיניסטיות במנוע פיזיקלי הצורות אשר הושפאו ע"י צורות אחרות לא התנהגו שווה בקליינטים שונים, בתופעה הזאת היה ניתן להבחין רק בזמן משחק רב משתתפים כאשר בזמן הפגיע של אובייקט באובייקט אחר, התנוע של אובייקטים היתה שונה בין הקליינטים. זה גרם לכך שנאלצנו לבטל תנוע בחלק מהצירים, ובציר הZ בפרט.

## מערכת Events

לפי ה design של הפרוייקט, כל הרכיבים הלוגיים לא מודעים לקיום רכיבים ישיר אלה אם כן אובייקט מסויים מכיל אובייקט אחר תחתיו, אזי האובייקט אב יכיר את הבן.

על מנת להעביר מידע חיוני בין הרכיבים השונים לדוגמא: input ושחקן, כתבנו מערכת ארועים גנרית המבוססת על delegates .

כאשר רכיב מסויים מסיים לבצע פעולה כלשהי בעלת משמעות אפשרית לרכיבים אחרים, הוא מרים ארוע מתאים וכל רכיב שרוצה לפעולה בתגובה לפעולה זו, אחראי להזין לארוע זה ולפעול בהתאם. על מנת למנוע מרכיבים לא פעילים להזין לארועים, ההרשמה להאזנה מתבצעת אך ורק כאשר הרכיב הופך לפעיל.

## State machines

כמעט כל רכיב במשחק משתמש במכונת מצבים אסינכרונית לניהול המצבים הפנימיים שלו על מנת לגרום לקוד ליהיות מסודר יותר ללא שימוש בדגלים ועל מנת להפחית את סיבוכיות הקוד.

כל רכיב שרוצה להשתמש במכונת מצבים, צריך לממש Interface מתאים ולממש את המתודה

SetState. כמות המצבים ושמם נקבעים על ידי enum המוגדר בכל רכיב הממש מכונת מצבים.

## MVC

בכדי לבנות את המשחק השתמשנו בכמה כלים ושיטות תכנות, בינייהן מודל ה MVC .

במודל זה ישנם שלשה דברים האחראיים על תפקוד צד השרת והקליינט במשחק :

MODEL : המודל אחראי על הפונקציונליות של ישות מסויימת – במודל מוגדרות פונקציות הכרחיות לאותה היישות כגון הבאת נתונים מXML , שמירת נתונים, הגדרות אתחול וכו'.

VIEW :הVIEW אחראי על מה שיוצג ללקוח – זהו המקום לממש את התצוגה של המשחק.

CONTROLLER: הקונטרולר אחרי על לגשר בין המודל והVIEW.

הקונטרולר יבצע הבאה של נתונים ממסד הנתונים(לדוגמא) ויטען את הVIEW המתאים למצב הנוכחי של המשחק תוך שליחת הנתונים הרלוונטים לVIEW.

## Matchmaking

מערכת המטצמייקינג היושבת בצד שרת אמורה להבטיח את איכות המשחק עבור שחקנים מנוסים אשר מצפים לקושי ואתגר מהמשחק. שחקנים תמיד מקבלים את השלב המתאים לרמה שלהם, מנגנון זה מבטיח אתגר מתאים לכל רמת משחק.

בעת החיבור לצד שרת, השרת יודע לנתח את הדירוג של השחקן, לפי הדירוג זה השחקן יצורף לחדר קיים איכןה שממתין לו שחקן בעל רמה דומה, או יפתח חדר חדש לצורך המתנה לשחקן מתאים מבחינת רמה ונסיון.

אם אחרי פרק זמן של 10 שניות לא נמצאית התאמה בין השחקנים ברמה מסויימת, רמת ההתאמה תופחת לרמה בינונית, יתבצעה חיפוש חדש בחדרים קיימים ואם שוב לא תמצא התאמה, השחקן יכנס ל-10 שניות נוספות. אם לאחר 10 שניות נוספות שוב לא נמצאת התאמה, רמת ההתאמה תרד לרמה נמוכה. וכדומה עד לביטול ההתאמה בכלל.מנגנון זה מבטיח שלא יקרה מצב בו שחקן מחכה זמן רב מדי ללא מציאת התאמה.

## Custom gestures detection

כחלק מכתיבת הפרוייקט, נתקלנו בבעיה שתנועה מאד חשובה למשחק לא קיימת ב SDK זיהוי התנועה של Microsoft עבור Kinect, התנועה היא Grip או זיהוי יד מכווצת לצורך איתור תפיסת אובייקט.

הפתרון שמימשנו כולל מלבן מסביב ל joint של כף היד, בכל פריים מחושב אחוז המילוי של המלבן על ידי חישוב על Grayscale ומציאת אחוז הפיקסלים הצבועים, בצורה כזו ניתן לבדוק בדיוק די גבוה האם היד של השחקן פתוחה או סגורה.

## Object oriented design

תכנות מונחה-עצמים או לעתים תכנות מוכוון-עצמים  היא פרדיגמת תכנות המשתמשת ב"עצמים" (אובייקטים) לשם תכנון תוכניות מחשב.

מרכיב בסיסי בתכנות מונחה-עצמים הוא המחלקה (Class). מחלקה היא מבנה מופשט בעל תכונות (Properties) המגדירות ומאפיינות את המחלקה כלפי חוץ, ופעולות (Methods), שהן פונקציות ייחודיות למחלקה. בחלק משפות התכנות קיימים גם אירועים (Events), שהם שגרות השייכות למחלקה ומוזנקות בהקשרים שונים, למשל בתחילתו או בסיומו של הליך או כתגובה לקלט מהמשתמש.

בפרוייקט הנ"ל השתמשנו בקונספטים הבאים של תכנות מונחה עצמים:

* Classes
* Structs
* Methods
* Events
* OOP patterns
* Interfaces and abstraction

## Source control

על מנת למקבל את העבודה שלנו על הפרויקט השתמשנו במערכל לניהול גרסאות הנקראת Git

אשר אפשרה לנו לעבוד במקביל, מבלי לפחד שהשינויים של אחד ידרסו את השינויים של השני, השתמשנו ב מערכת Assembla ([קישור](http://assembla.com/)|) בשביל אירוח ואחסון הפרוייקט.

בנוסף לכך המערכת מאפשר לעקוב אחרי logs וטבלת שינויים קודמים בקלות ובנוחות.

# בדיקות, תוצאות והערכה

## עמידה ביעדים שנקבעו

בסך הכל אנו מגדירים את העבודה שהתבצעה כמוצלחת, השגנו את רוב המטרות שנקבעו בתחילת הפרוייקט.

### לאפשר משחק Cross platform יחודי בין מכשירי מובייל, מחשב אישי וקונסולות

לדעתנו יעד זה הושג בהצלחה, הצלחנו לגרום למשחק לרוץ על מספר פלטפורמות רב ביניהן:

* מחשב אישי בסביבת חלונות
* מקינטוש
* אייפון
* אייפאד
* אנדרואיד
* WP8

אנו מרגישי שהשקענו עבודה רבה ביעול ושיפור ביצועים על מנת שהמשחק ירוץ על מכישירי מובייל בסטנדרטים שונים.

### רצון לייצר משחק חדש המשלב מספר זאנרים פופולריים מתחום משחקי המחשב

אני מרגישים שהצלחנו ליצור proof of concept ראוי לכל דבר אשר מציג את הכוונות שלנו בהגדרת הפרוייקט ומייצג את האפשרות ליצור משחק מהסוג שהוגדר באפיון. כמו כן מהבדיקות שביצענו בקרב קהל testers פנימי (חברים ומשפחה), נוכחנו לראות אכן נהנים לשחק במשחק שלנו ומסכימים על כך שיש לזאנר משחקים חדש זה פוטנציאל גבוה שעדיין לא מומש בשוק המשחקים הבינלאומי.

### **התקמצעות בתחום הReal time client – server multiplayer**

תחילה ניסינו לכתוב את צד השרת (Server side) בשפת תכנות הנקראית PHP, ללא שימוש בframeworks נוספים וללא תלות ב skd's כלשהם. אך במהרה הבנו שמדובר במשימה גדולה עבור scope הפרוייקט הנוכחי, במיוחד לאור העובדה שהפרוייקט כולל המון פיצרים נוספים בנוסף לצד שרת.

חקרנו מספר אופציות של צד שרת עבור משחקי Real time מולטיפלייר שבהם חשוב מאד זמן התגובה וכמות העדכונים בשניה. בין האופציה שנבחנו היו:

* שרת היושב אצל אחד השחקנים
* שרת יעודי המבצע ניתוב בין השחקנים
* שרת יעודי המכיל מנוע פיסיקלי ומעדכן את השחקנים בכל פריים

בסוף בחרנו לפצל את צד השרת ל 2 חלקים:

* Matchmaking server – בחלק זה מתבצע חיבור בין השחקנים כאשר השרת משמש כשרת Authorative עבור שני השחקנים.
* Game server – בשלב זה השרת משמש רק לניתוב האינפורמציה העוברת בין השחקנים ומגדירה את מצב הנוכחי של המשחק, השרת אינו מבצע חישובים מסדר גבוה ולכן מספר התקלות שיכולות להיווצר הינו נמוך מינורי ביחס לפתרונות האחרים שצויינו.

בחרנו להשתמש בframework (ספריית עזר) הקיים בשוק הנקרא Player.IO, אשר הוכיח את עצמו במשחקים קודמים שבנינו במסגרת קורסים אחרים בשנקר. במהלך ניסיוננו הקודם עם ספריה זו, ראינו שהיא נוחה לשימוש, דינאמית, ידידותית למתכנת וכמעט ללא באגים ותקלות.

תהליך ה deploy עם מערכת זו הינו פשוט ונוח וכולל אך ורק הוצאת DLL והעלאתו לשרת יעודי.

הDLL מכיל בתוכו את הקונפיגורציה של צד השרת ומימוש לוגיקת שרת עבור המשחק שכתבנו בהתבסס על ספריית העזר של Player.IO.

מערכת ה Matchmaking מיושמת גם היא בספריות של Player.IO.

### להתנסות ביצירת Gestures חדשים עבור מצלמת הקינקט

אומנם התנסנו בכתיבת זיהוי תנועות חדשות בממשקי זיהוי תנועה, אך לדעתנו לא השגנו את המטרה בצורה מלאה מכיוון שלא הצלחנו להביא את את מנגנון זיהוי Grip לצורה ראויה להדגמה. אנו מרגישים שאם היה לנו זמן פיתוח נוסף, היינו יכולים להביא את המנגנון למצב פעיל בדמו אך בחרנו להתמקד במטרות היותר עיקריות של הפרוייקט.

הטכנולוגיה איתה עבדנו (Kinect) אינה מתאימה ב 100% לזיהוי תנועות מינוריות, ייתכן והיינו צריכים לבחור מלכתחילה מצלמת Close range כדוגמת המצלמה של Intel-Creative.

### בניית פרוייקט בקנה מידה מקצועי בתחום משחקי המחשב

אנו חושבים שהפרוייקט שכתבנו הינו בקנה מידה גדול יותר מכל פרוייקט אחר אשר ביצענו במהלך הלימודים.

שילבנו מספר מערכות לכדי פרוייקט יחיד והתנסנו באינטגרציה של רכיבים חיצוניים, האינטגרציות כללו את הרכיבים הבאים:

* שליטה על ידי מסך מגע (טאץ)
* שליטה באמצעות מצלמת Kinect
* רכיבים גרפיים שנקנו ב Asset store של Unity
* Plugin עבור תקשורת עם צד שרת
* מערכת GUI

## Code complexity analysis

במהלך הפרוייקט, ערכנו מספר פעמים בדיקות סיבוכיות קוד ( Code metrics ) על מנת להישאר בגבולות המקובלים בתעשיה, להלן רשימת הפרמטרים שנלקחו בחשבון:

* Cyclometric complexity
* Class coupling
* Code size
* Comments size

## בדיקות איכות תוכנה

על מנת להבטיח את איכות המשחק, כתבנו 2 מסמכי STP בסיסיים שביצאנו על מנת לוודא שאין באגים חמורים הפוגעים בפונקציאונליות. התסריטים בוצעו על שלושת הפלטפורמות העיקריות של המשחק PC, iOS, Android.

סיכום צעדי בדיקות צד לקוח:

* פתיחת משחק
* נויגציה בין כל המסכים ושימוש בכפתור Back
* התחברות לשרת
* עם תקשורת
* ללא תקשורת
* עם חיבור לא יציב
* התחברות לשרת
* יצירת חדר
* התחברות לחדר קיים
* יציאת שחקן באמצע והשפעת המהלך על שחקן שני
* משחק יחיד
* וידוא ניקוד נכון בהתאם לביצועים

## Performance analytics

* בדיקה ומעקב אחר קצב ה Frames per second
* הקצאת זכרון מירבית
* ריצה לזמן ממושך ווידוא אי זליגת זכרון
* בדיקת ניצול CPU
* ניתוח Draw calls פר אובייקט

## ניתוח סטטיסטי לבניית שלבים

במהלך המשחק מתבצע תהליך איסוף נתונים בזמן ארועיים קריטיים, בסיום כל Session , הארועים נשלחים לשרת ונשמרים בבסיס נתונים לצורך מעקב וניתוח.

להלן רשימת הפרמטרים המדווחים:

* תוצאת משחק
* רמות השחקנים
* כמות פגיעות

אנו מאמינים בעזרת מידע חיוני זה ופרמטרים נוספים ניתן לכוון את רמת הקושי של השלבים על מנת להתאים את השלבים בצורה מירבית לרמת שחקן.

# דיון: מסכנות ופיתוחים עתידיים

## תכנון נכון על מנת לעמוד ביעדים

במהלך הפרוייקט הרגשנו לחץ רב ולעיתים חששנו שלא נספיק לסיים אותו בזמן, להערכתנו התכנון של הפרוייקט היה אופטימי ושאפתני יתר על המידה ולא התאים לscope הזמנים שהוקצב.

## מולטיפלייר קבוצתי

אחת המטרות האופציאונליות ששנרשמה במסמך האפיון היא לאפשר מולטיפלייר תחרותי בין קבוצות בנות שתי שחקנים כל אחת, סך הכל ארבעה שחקנים במשחק.

בתחילת המשחק, שחקנים מחפשים פרטנר או מזמינים חבר מסויים להיות הפרטנר ויוצרים קבוצה מהסוגים הבאים:

* Premade – שחקנים המכירים אחד את השני.
* Randomly picked according to ratio- שחקנים אשר נבחרים רנדומלית.

לאחר מכן מבצע תהליך Matchmaking אשר מוצא לקבוצה המגובשת קבוצה יריבה ברמת קושי השווה לממוצא את רמות של שתי השחקנים וזאת על מנת להבטיח אויבים שווי ערך.

לאחר מציאה מוצלחת של קבוצה יריבה, שני הקבוצות מועברות למסך המשחק השונה ממסך משחק רגיל באופן הבא:

* חלוקה לשתי חלקים – כל חלק של מסך שייך לקבוצה מסויימת
* אותות השחקנים – כל קבוצה מקבלת צבע לאותות שחקניה לצורך זיהוי פשוט של הקבוצות

עם סיום שלב זה, המשחק הופך להיות לזהה למשחק הרגיל למעט שלב החישוב הניקוד הסופי אשר בו מתבצעת השווה בין ניקוד הקבוצות ומתבצעת הכרזה על הקבוצה המנצחת.

למצב משחק זה יוקצה Leaderboard חדש ויערכו טורנירים חודשיים\שבועיים\יומיים בהם קבוצות בעלות מספר נצחונות רב ביותר יכנסו לטורניר אלמנציה והקבוצה שתזכה בגמר תקבל פרס גבוה.

## השמה למטרות חינוכיות

המשחק מתוכנן להיות משחק המפתח אסטרטגיה וקריאטיביות בקרב ילדים, המשחק בנוי בצורה כזאת שממש כל להבין את כללי המשחק וכמו כן ניתן להוסיף סוג משחק המלמד אותיות וספרות ע"י הגדרת משימה שהיא להגיע לצורה שהיא אות או מספר, כך שהשחקן אמור לבנות את הא"ב או ספרות 1-10.

אנו מתכננים להפיץ את המשחק לבתי ספר וגנים ממגזרים שונים ולתת את האפשרות להתאים את קונספט המשחק למטרות חינוכיות של אותו מוסד.

## התאמת המשחק לאנשים בעלי מוגבלויות

השליטה במשחק מאפשרת לאנשים בעלי מוגבלויות להשתתף במשחק ולא להרגיש פחות טובים מיריבם "הרגילים", בשביל לשלוט במשחק מספיק נגיעות כלילות במסך, או עכבר, ללא כל צורך במקלדת.

הצעדים אותם ננקוט על מנת לגרום למשחק להיות נגיש לאנשים בעלי מוגבלויות:

* שליטה קולית
* שליטה בעזרת ג'ויסטיק יעודי הדורש שימוש רק בעצבה אחת
* התאמת צבעים לאנשים בעלי אברון צבעים
* פישוט הטקסטים

# מקורות

* Unity scripting reference
* Unity Game Development Essentials by Will Goldstone Will GoldstoneWill Goldstone
* Game Coding Complete, Fourth Edition by [Mike McShaffry](http://www.amazon.com/Mike-McShaffry/e/B002YGQ3RW/ref=ntt_athr_dp_pel_1)
* Game Engine Architecture by Jason Gregory
* The C++ Programming Language (4th Edition) by Bjarne Stroustrup
* [Object oriented programming](http://en.wikipedia.org/wiki/Object-oriented_programming)
* Pro C# 2010 and the .NET 4 Platform by Andrew Troelsen
* Meet the Kinect: An Introduction to Programming Natural User Interfaces by Jonathan C. Hall & Phoenix Perry
* Microsoft Kinect SDK reference on MSDN
* Algorithms and Networking for Computer Games by Jouni Smed

# נספחים

* מסמך FRS
* הצעת הגשה
* מצגת הצגה

# תוכנה/דמו מוצר