Snake game

**לפני המשחק:**

יש לאתחל קודם המשחק בקובץ utilities את הנתונים כפי שרוצים שיהיו במשחק, למשל גודל חלון, מהירות, מספר אובייקטים לאכילה בשביל לעבור שלב.

**תיאור ומהלך המשחק:**

*תזוזה-*

הנחש זז ימינה באופן דיפולטיבי.

השחקן ישתמש במקשי החיצים לכוון את הנחש.

*עצירת המשחק-*

מקש רווח משהה את המשחק ואת השעון, לחיצה נוספת תמשיך את המשחק.

בהתנגשות בזנב או בגבולות החלון ייפסל השחקן ויוצג לו הניקוד.

בלחיצה על play again הניקוד יאופס והמשחק יתחיל שוב.

*אוכל הנחש-*

תות/מטבע. הנחש גדל באכילתם.

*ניקוד-*

התות מגדיל ניקוד באחד. המטבע מופיע ונעלם ויותר קשה לאכילה, לכן מגדיל ניקוד ב 3.

*שלבים-*

מעבר שלב לאחר אכילת כמות מסויימת תלוי בשלב (הוגדר בקבועים)

בשלב העוקב- יגדל החלון, ישתנה הרקע ומספר השלב, ומהירות הנחש תגדל. עקרונית ניתן להוסיף שלבים מיידית ע"י הוספת טקסטורה בלבד או אפילו בלי. אנחנו כרגע הגבלנו לארבעה שלבים.

*גרפיקה-*

נציג בחלון- תמונת רקע, מספר השלב, ניקוד, שעון-זמן משחק, בהשהיה נציג pause ובפסילה נציג את הניקוד שנצבר וכפתור play again.

*אנימציה-*

לכל אובייקט יש אלגוריתם משלו לאנימציה לפי מספר הספרייטים שיש לו.

פה הנחש נפתח ונסגר.

התות והמטבע מסתובבים.

*אודיו-*

שיר רקע למשחק, צליל באכילה, צליל U WIN"" בסיום שלב.

**תיכון:** מצורף קובץ UML.

*דגשים:* התיכון ייחודי בכך שהקפדנו בצורה מוחלטת על עיקרון ה- encapsulation

הדבר בא לידי ביטוי ב-

1. יצירת אובייקט מיוחד collider שנוצר יחד עם כל אובייקט

המצריך טיפול בהתנגשויות. אובייקט ה- collider מחזיק בתכונה checkCollision

אשר תבדוק התנגשות ללא גישה לאובייקטים עצמם רק דרך ה- collider אותו הם

מחזיקים.

1. ה- controller שמחזיק את חלון המשחק, לא יקבל גישה לאובייקטים של המשחק בכדי

לצייר אותם. רק נעביר מצביע לחלון ל - GameModeאשר מחזיק את האובייקטים

והוא יכניסם לחלון.

1. ה- controller הוא בעצם תבנית לניהול כל משחק. הוא רק מחזיק מחלקות מהסוג MODE ע"י

מחסנית והם אלו שמכילים את המשחק, ואיננו ייחודי רק למשחק זה, בכך אנו מאפשרים בקלות

הוספות ושינויים למשחק עצמו ללא צורך בעידכון ושינוי ה- controller.

מחלקת controller מנהלת את המשחק. מכילה את מחלקת gameMode שמחזיקה את כל האובייקטים ואחראית עליהם.

gameMode מכילה ראש (שמכיל גם את הזנב (בוקטור של זנבות) ואחראי לו) אוכל מתנות וכפתורים. כל האובייקטים האלו יורשים ממחלקת gameObj.

מחלקות אוכל ומתנה יורשות ממחלקה EATABLE שממנה יורשים האובייקטים הניתנים לאכילה. לכל מחלקה יש אלגוריתם שלה של ציור (בהתאם לכמות האנימציות).

מחלקות gameMode lostMode יורשות ממחלקה Mode , כל מחלקה בעצם מחזיקה מצב משחק כלשהןא (משחק, הפסד)

controller בלולאה ראשית מוחקת את הלוח. בפונקציה updatePos קוראת gameMode שבודקת אם לעצור משחק או לסיים בגלל פסילה או סיום שלב. אם לא, מזיזה את הראש והאוכל למיקום החדש בהתאם לשינויים (תזוזה מהמשתמש, אכילה של אוכל ישן באוכל).

בפונקציה process היא קוראת gameMode שמשנה מה שצריך בהתאם לשינויי מקום שבוצעו ובודקת אם היו התנגשויות ואיזה. במידה והיו התנגשויות היא מעדכנת את הנצרך.

.

**מחלקות נוספות:** collider אחראי לבדיקת התנגשויות. כל אובייקט במשחק (ממחלקת gameObj) מכיל אותו, הוא מקבל collider של אובייקט אחר ומחזיר אם הייתה התנגשות.

מחלקת utilities מכילה את נתוני המחשק והחלון.

**אלגוריתם מעניין:** בדיקת ההתנגשויות נעשית ע"י מחלקה מיוחדת שקיימת בכל אובייקט (collider) והוא רק בודק האם האובייקט הנוכחי מתנגש ב collider אחר. את השינוי והעיבוד תעשה המחלקה המנהלת בלבד.

בתזוזת הנחש הלוגיקה היא רק בהזזת הראש. לאחר מכן כל חלק בזנב מקבל את מקום החלק שלפניו כך התזוזה נהיית פשוטה מאוד.

**מבני נתונים:** עבור מצבי המשחק השתמשנו במחסנית כדי שנוכל להוציא מצב כלשהוא שהסתיים (ע"י פופ). המחסנית מתרוקנת ונטענת מחדש במצבי המשחק כשהמשחק מסתיים (ניצחון או הפסד). עבור הזנבות השתמשנו בוקטור.

**תבניות עיצוב:** המחלקה Utilities מחזיקה את הקבועים ונתוני המשחק, המחלקה היא Singelton. כיוון שנתוני המשחק והחלון צריכים להיות גלויים לכול המחלקות בקוד ומצד שני הם משתנים ולא קבועים.

הקוד נקי מאוד ואין כמעט מעבר נתונים בין מחלקה למחלקה.

כל הקוד מתועד במקום.