PROJECT REPORT – WEEK 1 Necati Çevik-1746551

Market Research

I looked for softwares that share similarities with our project's concepts and found out that our project is at it's core, is a programming game. The wikipedia article about the programming game describes the concept as;

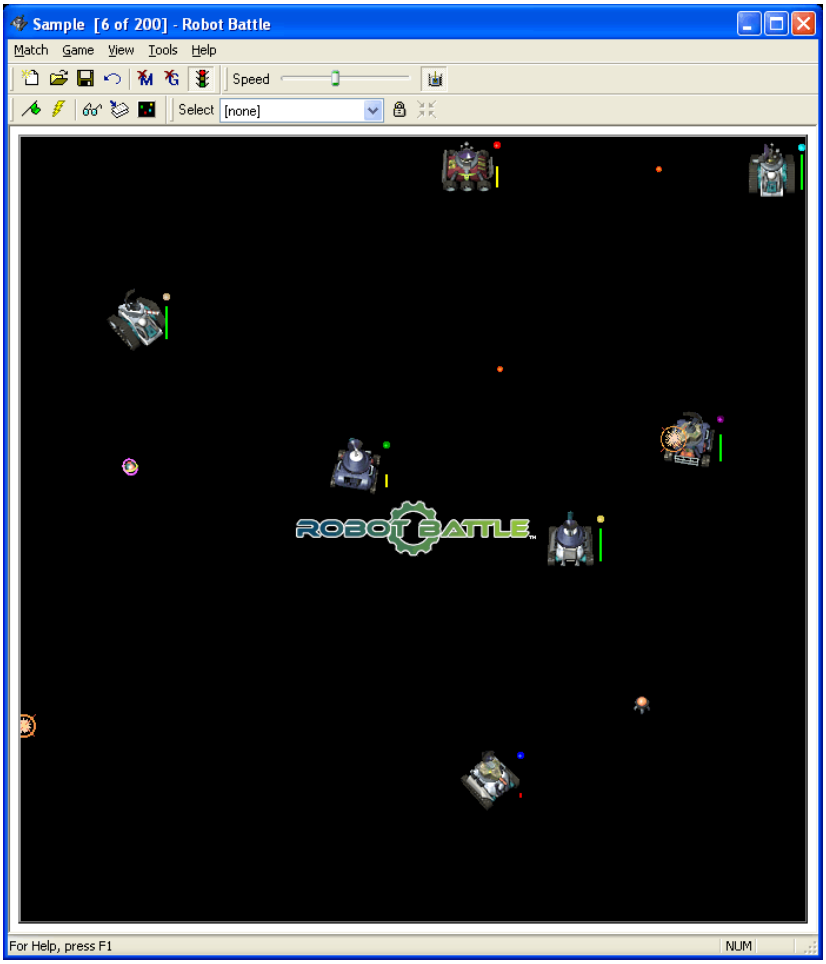
A **programming game** is a computer game where the player has no direct influence on the course of the game. Instead, a computer program or script is written in some domain-specific programming language in order to control the actions of the characters (usually robots, tanks or bacteria, which seek to destroy each other). Most programming games can be considered environments of digital organisms, related to artificial life simulations.

A few programming games of note include *RobotWar*, *Core War*, *Mouse Run* and *RoboCode*.

There are different tournaments and leagues for the programming games where the characters can compete with each other. Usually a script is optimized for a special strategy. For instance, in *Mouse Run*, a Java class is written by a programmer, which will provide the logic for a mouse to navigate through a maze and compete with other mice to find and consume a cheese first.

Programming games may be derived from almost any other type of game. For example, the World Computer Chess Championship consists of matches between programs written for the abstract strategy game of chess. Also, some non-computer games borrow elements of programming games; in the board game RoboRally, players arrange cards to "program" their pieces with a sequence of moves, causing moves to be made without the knowledge of one's opponents' preceding moves.

As I searched for more programming games I came upon a particular game called ***Robot Battle*** which was released in 2002 for Microsoft Windows. Here is a screenshot from the game.



Home site of the project describes the above photo as; seven robots battling for supremacy. Robot skins and background images are fully customizable (large graphics size shown).

The wikipedia article about the game describes the games as;

Robot Battle is a programming game for Microsoft Windows where players design and code adaptable battling robots. Robot Battle takes strategy rather than reflexes, accuracy, or timing to succeed. What differentiates one robot from the next is its programming, for which the player is responsible. The game is inspired by the similar game *RobotWar*.

I noticed that they defined their Robot as;

The robots in Robot Battle have three separate parts. The body contains the tracks, is a square size 33x33, and rotates at a speed of 5 degrees per turn. The gun has the ability to shoot energy missiles which will damage robots, and destroy missiles, mines and cookies on collision, and can rotate 10 degrees per turn. The radar has the ability to scan for cookies, mines, robots, and walls, and can rotate 15 degrees per turn.

This definition of the robot is quite like we defining the rules of our football game or the physics rules that will be applied during the simulations.

The robots of this game are coded in Robot Scripting Language. A brief information about the language can be found again in the wikipedia page of the game.

Robots are coded in **Robot Scripting Language** (**RSL**), and can be created in all text editors such as Notepad. Robots are usually worked on and distributed in .prg format, but some coders choose to scramble their robots and distribute them as .dst files. The scrambling software was created by Brad for Robot Battle 1.3 and later edited by Joseph Fowler (aka Sorcerer) and Mark Duller to give two separate programs both compatible with Robot Battle 1.4.

RSL has been said to resemble many programming languages including Basic, C, and JavaScript. Each robot has one or many sections, separated by section names and curly brackets, "init" being the only required section and where other sections are defined by an event and a priority.

The Core section does not need a priority, and contains the details of what the robot will do when it isn't doing anything else. The Ascan, when used, usually has the lowest priority (least important) and takes place whenever the robot is moving. The other events take place when an object is detected with the radar or collided with by the body.

The parts of the robot are controlled by commands.

The easiest way to make a robot is to look at the ones which come free with the game, *Combo*, *Smart Corner*, *Events*, *Fire*, *Rammer*, *Shell*, *Side Liner*, *Target*, *Walls II* and *Zag*. These robots demonstrate uses of the simple commands above, as well as some more advanced tactics such as the use of radio and the differences when creating a robot with "command blocking" turned off.

Their approach differs from ours here because we will give a detailed API and an easy to implement Java interface to our users. While a game specific language such as RSL may have it's advantages, I think that we are better off with java. The reasoning behind this is that the players of the game have to learn the new language to play the game. I think that the learning process might set off some potential players.

As a conclusion, I learned about the concept of a programming game and this will help me while I am designing the parts of the project.