1. Introduction

1.1 Purpose of the system

Farmio is a farming simulation game. It has implementation of different features that are actually implemented on real-world farming. There are tools to take care of the plants, different seeds and some other implementations of real farms. As we examine, we noticed that there are lots of objects in the real world farming and they have some connection between them. Therefore, in the step of looking and trying to implement it inside computer is better and easier.

1.2 Design Goals

Before the composing the system it is crucial to identify the design goals of the system in order to clarify the qualities that our system should focus on. In this respect many of our design goals inherit from non-functional requirements of our system that are provided in analysis stage. Crucial design goals of our system are described below.

End User Criteria:

Ease of Use: Since our system is a game, it should provide good entertainment for the player. In order to provide the entertainment player should not have a difficulty in using our system. In this respect, system will provide player friendly interfaces for menus, by which player will easily find desired operations, navigate through menus and perform the desired operations. While a player is “farming” , speed will be important. If the seeds grow too slow, player might be bored or if too fast again the player might feel that what he has done is not satisfiying. Also it will be important to have great graphics to keep player interested. It is determined that our system will perform actions according to mouse input from the user, like clicking buttons, moving around the farm. This makes it easy to use the system from the point of the player.

Ease of Learning: Since player is not ought to have knowledge about how the game is played, how to plant a seed, how to water and grow a seed. It is vital for the user to obtain information about the game concepts, for this purpose system will provide an instructive help document, by which he will be easily get warmed up to the game. Also there will be a litttle instruction set in the begining of the game so the player can learn hands on.

Maintenance Criteria:

Extendibility: In general, in the lifetime of game software, it is always important to add new components, features to the game in order to sustain the excitement and interest of the player. In this respect our design will be suitable to add new functionalities, entities (i.e. new brick types, new power-ups) easily to the existing system.

Portability: Portability is an important issue for a software, since it provides that the software can reach wide range of users. In this respect we are determined that the system will be implemented in Java, since its JVM provides platform independency, our system will satisfy the portability.

Modifiability: In our system it would be easy to modify the existing functionalities of the system. In order to achieve this we will minimize the coupling of the subsystems as much as possible, to avoid great impacts on system components by a desired change.

Performance Criteria:

Response Time: For the games, it is vital that users’ requests should be responded immediately in order not to distract the player’s interest and entertainment. Our system will respond player’s actions almost immediate, while also displaying animations, effects smoothly for enthusiasm.

Trade Offs:

Ease Of Use and Ease of Learning vs. Functionality:

In our system we determined that player should be able to learn and use the system very easily. Therefore our design proposes that the priority of the usability is higher than functionality. In other words our system does not bother the user with complex functionalities or we do not make the user to be lost in many functionalities, in order to make our system easy to understand and use.

Performance vs. Memory:

In our system, it is our main purpose to make the animations, effects, transitions very smooth. Briefly, performance of our system is our primary focus. For this purpose, we sacrificed the memory in order to gain the performance. For instance: in our game we have a bonus called explode explosive bricks, within the regard of this bonus we should immediately explode all explosive bricks on the map. Since the performance of this event is important in our system, instead of iterating the all map for explosive bricks, we store the important game objects of the game map in memory to access them fast when needed.

// buradayım

1.3 Definitions, acronyms, and abbreviations

Abbreviations:

MVC: [2] Model View Controller

JDK: [1] Java Development Kit

JVM: [1] Java Virtual Machine

1.4. References

[1] http://en.wikipedia.org/wiki/Java\_(programming\_language)

[2] Object-Oriented Software Engineering, Using UML, Patterns, and Java, 3rd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2010, ISBN-10: 0136066836.

1.5. Overview

In this section, we represented purpose of the system, which is basically entertaining the player as much as possible, to achieve this purpose we defined our design goals in this part. Our design goals are determined according to provide the portability, ease of use, ease of learning, high performance, high maintainability. In this respect of course we made some trade-offs to realize our goals. We sacrificed from functionality to make our system simpler and understandable, also we sacrificed from memory to gain performance on playing smooth animations and effects.