Bilkent University
Department Of Computer Engineering

`

Senior Design Project

Project short-name: AugCards

Specifications Report

Yusuf Avcı, Burak Mutlu, Çerağ Oğuztüzün, Yiğit Görgülü, Bora Kurucu

Supervisor: Prof. Dr. Uğur Güdükbay
Jury Members: Asst. Prof. Dr. Can Alkan, Assoc. Prof. Dr. Çiğdem Gündüz Demir
Innovation Expert: Prof. Dr. Veysi İşler

Progress/Specifications Report
October 12, 2020

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Contents

Specifications Report

Project short-name: AugCards

1 Introduction

Mobile games are becoming more popular day by day [1]. With mobile gaming, a new era in the gaming sector has emerged. People started to lose their habit and passion of playing games with physical equipment, such as board games and card games. The emergence of mobile gaming, due to its appealing graphics, ability to play online, and the sky-high imagination of the mobile game developers, lead people to quit playing card games physically. Additionally, card games have limited assets, rigid visuals, and static rules.

The main philosophy of AugCards is reuniting the tradition and old-school fun of playing card games with your friends sitting around the table, the dynamism of mobile games. AugCard gives users the freedom to create their own cards with their own assets, introduce their own animations, and specify their own game rules.

Imagine you and your friends sitting around a table and want to have a good time. AugCard helps to limit social isolation caused by individual gaming, and bring the people together to create a game. You and your friends would first open the AugCards Desktop application and create the game cards, the event triggers, game animations, rules (Figure 1). Even the complex rules can simply be introduced with the help of user-friendly design which makes use of flowcharts and such structures. After the game is complete, you and your friends can open the complementary AugCards mobile application and everyone can tune in to play the game you just created, a network is established among the table and multiplayer mode is enabled. The cards and the AR versions of the assets on the cards accompanied by animations are seen on the table by everyone looking through the cam of AugCards (Figure 2). If you are proud of the game you

have created, you can share it on the AugCards platform for other users to play, and play a game made by another user.

In the Specifications Report of AugCards, we intend to give information regarding the overall specification of the system. In this report, the reader can find a description of AugCards, similar products, constraints, and the requirements of the project.

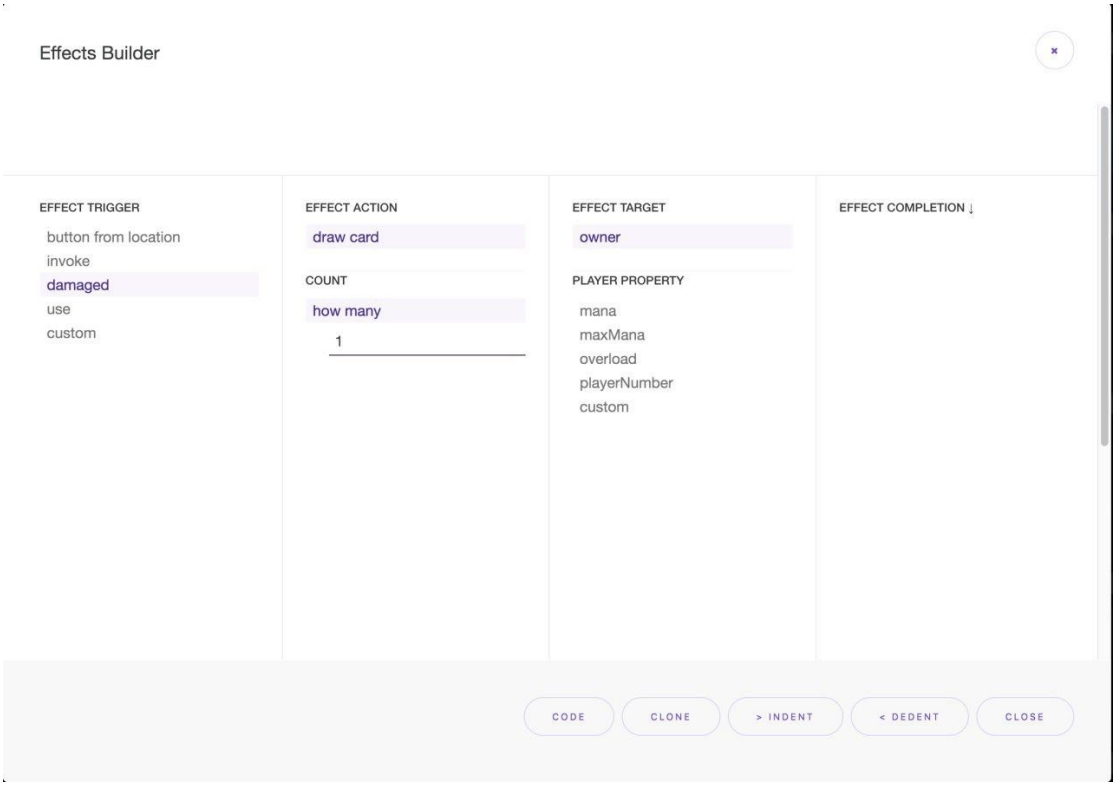

Figure 1: Sample Card Game making Screen from Dulst [2]

Figure 2: Sample Visual from an AR Yu-Gi-Oh Game [4]

1.1 Description

AugCards is a card game development tool where users can create their own cards and build custom games to be played on Android devices with multiplayer capabilities and AR supported visuals. The tool is innovative in the sense that it transforms the user-designed game logic and cards into AR multiplayer mobile games that can be played instantaneously by a group of friends around a table. Currently, there are engines that are aimed at making card games, however, there is no engine that supports mobile device integration and makes use of AR supported visual experience. A platform that brings together developers and players on this new era AR gaming is another innovative feature.

AugCards will also contain a cloud service where users can share the games they created for other people to see and play. AugCards eliminates the need to buy physical equipment for any card game and gives the user the opportunity to create and customize their own cards. Thanks to the AR system, the experience of playing a game will feel very similar to playing a game with a group of friends around a table, as the cards and effects will all be animated and displayed to the user. Furthermore, by using systems similar to flowcharts, we intend to make expressing complex rules of the game less of a challenge for creators. An example of complex

rules is the intertwined mechanics of Legends of Runeterra [5]. Our tool will overcome designing these complex rules. We will provide a custom diagram model which will help users design their game rules. An example for our diagram model can be seen in Figure 3.

The project will consist of two applications: one desktop application and one Android application. The game making part, where the user designs their own assets, cards, events, triggers, set of game rules, and card properties, will be provided via the desktop application. After the game is made and ready to be played, each user will use their mobile phones and use the complementary AugCards mobile app to see the AR visuals and cards on the table and play the game.

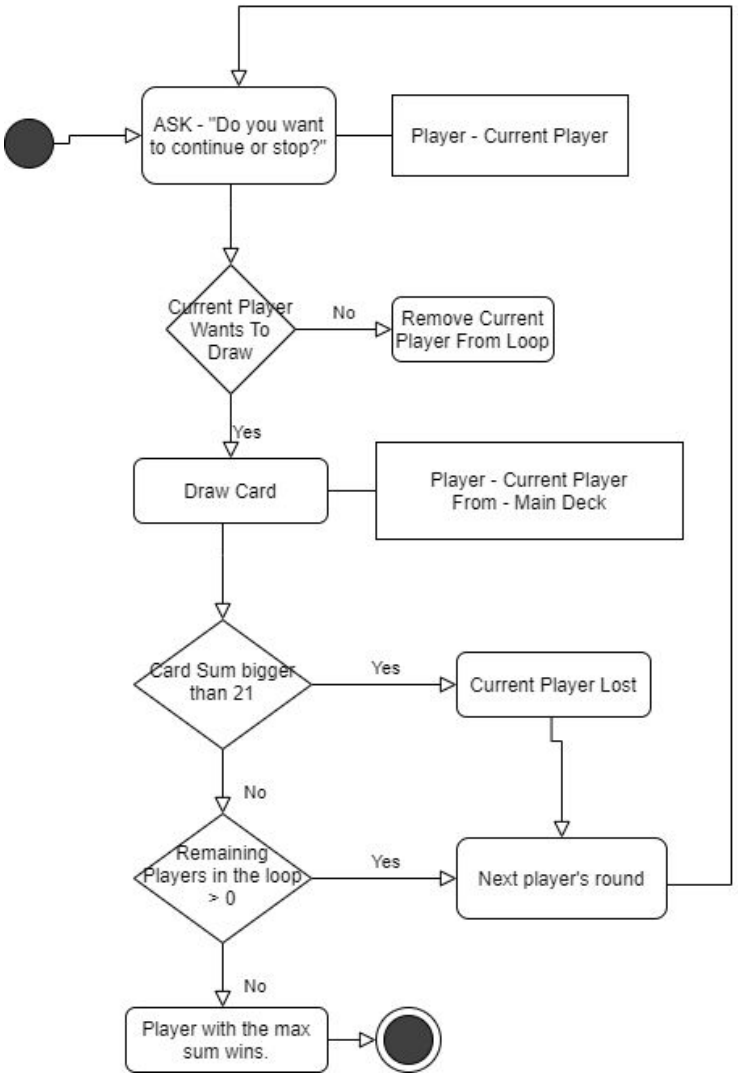

Figure 3: Custom diagram of game rules of Blackjack.

1.2 Similar Products and Technologies

Although there are many card web-based game-making platforms there isn't any mobile integrated one that specifically provides AR supported visuals and animations. Dulst is a free platform that provides users to create their own cards and play with them on the desktop [2]. Dulst also provides the users to share the games they have created and allows users to play with other users and review their games. We aim our project's desktop side to be an extended version of what Dulst has implemented. We aim to extend what Dulst, a platform with prominent success in this domain, achieved by adding mobile game and AR features to conform to the card game tradition of playing it with a group of friends sitting around a table.

Another game that aims to simulate VR experience is the Tabletop Simulator [6] which is a game that gives the user the feeling of playing on a table with friends, like our vision for AugCards. However, Tabletop Simulator does not support a mobile environment or AR graphics, since our game will be innovative and more complex compared to Tabletop Simulator.

1.3 Constraints

Following are the constraints of the project regarding the economic and development constraints.

1.3.1 Economic Constraints

Cloud service requires us to use servers, which may be highly costly. Most of the servers highly limit their usages if the cost is low, so we will have to search for one that will both satisfy the users in terms of gameplay, and us in terms of cost.

The domain of the website for AugCards may also require payment. Since most of the free domains lack some specific requirements, our team should find a suitable one or should be ready for buying domains.

1.3.2 Development Constraints

Project constraints are anything that restricts the development process of the project. These restrictions may be about cost, time, effort, memory, speed, communication, or any kind of limitation.

The game making part will be provided via the desktop application. After the game is made and ready to be played, each user will use their mobile phones. Learning different programming environments will take a considerable amount of time.

AugCards will also contain a cloud service where users can share the games they created for other people to see and play. This will take time and require us to learn how to write and manage servers, and probably will make us depend on the libraries that the programming language serves.

The games will be played using AR supported visuals. The choice of using AR over a regular GUI is especially challenging for development as all of us are new to this technology. Since using AR will highly consume memory and load highly on the CPU, this may result in a trade-off between game performance and speed.

1.4 Professional and Ethical Issues

This section of the Specifications Report is on the ethical and professional issues regarding the project implementation process.

1.4.1 Ethical Issues

AugCards should disable users from creating any type of card game which may include offensive content. The system should be able to check the contents of the user-created material and should be able to ban or restrict the users who constantly create offensive content.

AugCards should not store any kind of information that may violate the user’s privacy. The application should also ensure that users are not able to access other private information.

1.4.2 Professional Issues

The team should communicate via meetings and emails. Regular meetings will be conducted per week to discuss the current situation and future strategies. An email will be used to announce important decisions and meeting hours.

Every important decision and work will be provided in written format in order to prevent any kind of misunderstandings. The workload will be divided among team members equally.

The source code and GitHub repository will be kept private. This will increase the security of our work, and prevent any kind of plagiarism. Also, it is a necessity to actually profit from the game.

2 Requirements

2.1 Functional Requirements

2.1.1 User Functionality

- Users can be either game-developers or players.
- Users can search and look out for shared games on the platform.
- Users can add the games to their library by downloading binaries.
- Users can update the games on their library if creators release a new patch.
- Users can rate and comment on the games.
- Users can see the statistics of a game about downloads and ratings or read comments.
- Users should have an account to interact with the platform features.

2.1.1.1 Player Functionality

- Players can play downloaded games.
- Players can create and connect to private hosts for game sessions in the local network.
- Players should have the required number of human-players/mobile-devices connected to the host regarding the player constraints of the game.

- Players should set up the camera of the mobile device regarding that game content will be rendered on a planar surface.
- Players can send inputs to the game by touching the screen.
- Players can interact with game instances/cards and perform certain game events.
- Players can see other players' interaction and ongoing game events simultaneously.

2.1.1.2 Game-Developer Functionality

- Developers do not need programming knowledge.
- Developers can create custom card games.

By using UI options, developers should be able to define:

- required/custom game instances like the player, card, card set, card types, etc.,
- properties of game instances like player's health, card's properties in-depth (type, initial value, constant).
- class relationships between defined game instances.
- game events and their effects on game instances.
- trigger mechanism of game events.
- trigger mechanism of user-input events.
- rules for the events and the properties of game instances.
- heuristic function with accessible game data to enable AI scription.
- custom graphical models/assets for corresponding game instances.
- custom animations for each graphical model.
- bindings of each animation to certain events.
- Developers can upload/update the models for their own created games on the cloud.
- Developers can see all commits and pull an old version of the game.
- Developers can generate source code and compile the game on the cloud for Debugging or Release.
- Developers can generate binaries directly on the local machine for Debugging.
- Developers cannot access the generated source code for the game.

- Developers can download compiled binaries from the cloud.
- Developers can sync binaries of the game on desktop applications into a mobile application.
- Developers can debug the game with the desktop application over a virtual plane for AR-enabled graphics.
- Developers can debug the game with the mobile application over AR applied camera frames.
- Developers can deactivate certain game rules for easier debugging.
- Developers can share their custom game to make it accessible for users in the cloud platform.
- Developers can update/unshare their shared games.
- Developers can remove the repository of an unshared game from the cloud.

2.1.2 Desktop System Functionality

The system should:

- ask the developer for login credentials.
- create local/cloud repositories for custom game projects.
- delete all repository files if the developer decides to remove the game.
- restore the last committed version of a custom game from their cloud/local repository.
- restore a game over design models like game instances, instance relationships, events-triggers, etc.
- provide a sophisticated and interactive UI for each model design feature.
- provide a table design tool for game instances and their relationships such as inserting a new table to define new instance type, inserting a new line into instance tables to define properties, columns for the detail of properties like type, initial value, etc.
- provide a sequence diagram tool for events-triggers; definition of triggers contains determining check rules, updates for accessible properties, and invoking other events.

- provide a model viewer to import and bind graphical models/animations to game instances and events.
- give warnings about possible development issues like incomplete event-triggers, infinite-loop danger, etc.
- provide a commit tool to investigate commit details, push/pull commits, update game models over commits.
- provide a debugging tool to generate binaries on local/cloud, run the binaries on Android Emulator, or sync the binaries with the mobile application.
- provide an additional debugging feature to simulate the game with a desktop version to ensure quick debugging sessions.
- provide a platform interface to edit the community page for the game, share/unshare the game and interact with other users' comments.

2.1.3 Mobile System Functionality

The system should:

- asks the user for login credentials.
- provide a platform interface in which users can look out for custom games shared by developers.
- provide a community page view for each shared game, in which users can see and interact with stats/ratings of the game and comments by others.
- provide a download/remove button integrated within the community page to add the game to the library.
- provide a library to view the games installed on the user devices from both source Cloud and Desktop-sync.
- remove the games unshared by developers from the library.
- provide a game-sessions view to create/join hosts in a local network.
- provide a game-lobby view to see other players/devices connected to the host for a game session and launch the game session.

- provide an interactive in-game view to play the game; see the game content rendered on camera frames and send input to the game by touching the screen.
- include a common subsystem for all custom games, which performs the graphical/network operations on games.

2.1.3.1 Common Graphics/Network Subsystem

The system should:

- be a dynamic load library to ensure that downloadable/storable game binaries do not include it.
- provide a network functionality that any content update on a device should simultaneously appear on other devices connected to the same network.
- provide a graphics functionality that detects the planar surface from camera frames and render a custom content anchored to the surface on camera frames.
- provide motion detection to render the content with an accurate camera angle on each frame.
- support advanced custom contents like 3D/2D model/animations and UI elements.
- support advanced graphics effects like light-estimation, shadows, anti-aliasing, etc.

2.1.4 Cloud System Functionality

The system should:

- create private repositories for the game projects.
- store the game models along with commit logs in a private repository.
- generate source code and compile binaries for Debugging/Release.
- not store the debugging/release binaries.
- create public repositories to share games.
- store the last shared version (binaries) of the game in a public repository.

- store contents for the community page of the game in a public repository.
- provide access for developers to their private repositories.
- provide access for users to public repositories.
- provide file transfer feature to upload/download project commits, generated binaries, and shared games.

2.2 Non-Functional Requirements

2.2.1 Usability

- Game creation should not require programming knowledge.
- Tools should be self-explanatory, shouldn't require extensive tutorials or guides to be understood.
- The expression of complex card game rules will be simplified using flowcharts.
- AR-based graphics should have a refresh rate of at least 25 Hz to not affect game experience adversely.
- Popular image formats such as PNG and JPEG should be supported as assets.
- Popular graphical model/animation formats like OBJ and COLLADA should be supported.

2.2.2 Reliability

- Should ensure that changes in game models are not lost on connection errors.
- Should have a back-up mechanism for ongoing games in a network failure situation.
- Contradictory game rules shouldn't be allowed to cause errors.
- Cheats should be detected via checksums.

2.2.3 Maintainability

- Should be modular to reduce the complexity of the codebase.
- Network maintenance costs should be lower than 50TL per month while profits are low.

- Should use design patterns that will allow changing used libraries.

2.2.4 Accessibility

- Should be free to download.
- Should have integration with Google Services.
- Should require less than 1GB of RAM.

2.2.5 Extendability

- The addition of new possible game mechanics should not require changing existing code.
- The code itself should be properly structured,using design patterns and clever modularity.Adding new features and mechanics should require minimal or zero amount of change in the code.

2.2.6 Portability

- Should not cause any compatibility error with changing Android device sizes and camera resolutions.
- The game creation tool should run on Windows and Linux.

2.3 Pseudo Requirements

- Since the mobile application will be developed for the Android platform, the implementation language should be Java or Kotlin. For our project, we choose to use Kotlin because it is a modern language and is widely being used for Android development. We will use Android Studio for development.
- We will use Vuforia, which is a library to develop Augmented Reality applications for mobile devices. Vuforia supports Java, and Kotlin runs on the JVM which allows it to interact with Java libraries.
- To make integration with the mobile application easier and less problematic, the desktop application should also be written in Kotlin. For development, we will use IntelliJ IDEA.
- Since the projects will be hosted on a cloud service, we will need to interact with the chosen service's interface to use their cloud service. We

will use Firebase because it is free and developed by Google, which will help with integration with the Android application.

- We will license our project with the MIT License [7] since it provides the most flexibility and will be less likely to lead to legal problems we may come across in the future.

3 References

[1] "Topic: Mobile gaming." https://www.statista.com/topics/1906/mobile-gaming/ (accessed Oct. 09, 2020).

[2] "Dulst." https://dulst.com/ (accessed Oct. 08, 2020).

[3] *Yu-Gi-Oh! AR - Project A.T.E.M. - Attack & Defence Mode*. 2020.

[4] *Augmented Reality Tutorial No. 21: Unity3D and Vuforia for MultiTarget Tracking - YuGiOh! Card Game*. 2015.

[5] "Legends of Runeterra." https://playruneterra.com/tr-tr/ (accessed Oct. 09, 2020).

[6] B. Games, "Home." https://www.tabletopsimulator.com (accessed Oct. 09, 2020).

[7] "MIT License." https://mit-license.org/ (accessed Oct. 08, 2020).