Game Project Plan

“Illuminati”

CECS 343

Kevin Kobata
Kurt Tito
Jeffrey Heng
Calvin Vo

Introduction

Project Scope

Illuminati is a game for computer systems. A user can play a game of Steve Jackson Games' Illuminati alone or with multiple people locally. This game is built using the Unity3d game engine which will handle a majority of the display and calculation functions. Unity3d is also a graphical tool that will help us design and construct our card game, along with implementing card functions and user interactions using C# code.

Major Software Functions

Process and Control Functions

- Unity3d Engine: This is the main workhorse of the entire project. The engine will handle a majority of all functions, from generating the executable to graphical processing and user interaction

User Interface Processing

- Main Menu: This is the first interface that the player will encounter when first starting the game. From this menu, the player can check the manual for game, change settings, start a game, or exit the program entirely.
- Pause Menu: While in-game, the player can pause the game and either return to menu, check the manual, or quit the game.
- Help/Tutorial/Manual: These files are designed to help the user learn the rules of the game, whether outside of it, or while playing. This will cover the main game rules, how to use the program itself, and an FAQ covering some possible situations that may occur.

Input Processing

- User Input: This will allow the player to play the game through click and keyboard input.
- Databases: Unity3d game engine uses Visual Studios as a place to provide C# coding functions for each individual card and menu interactions within the card game.

Output Processing

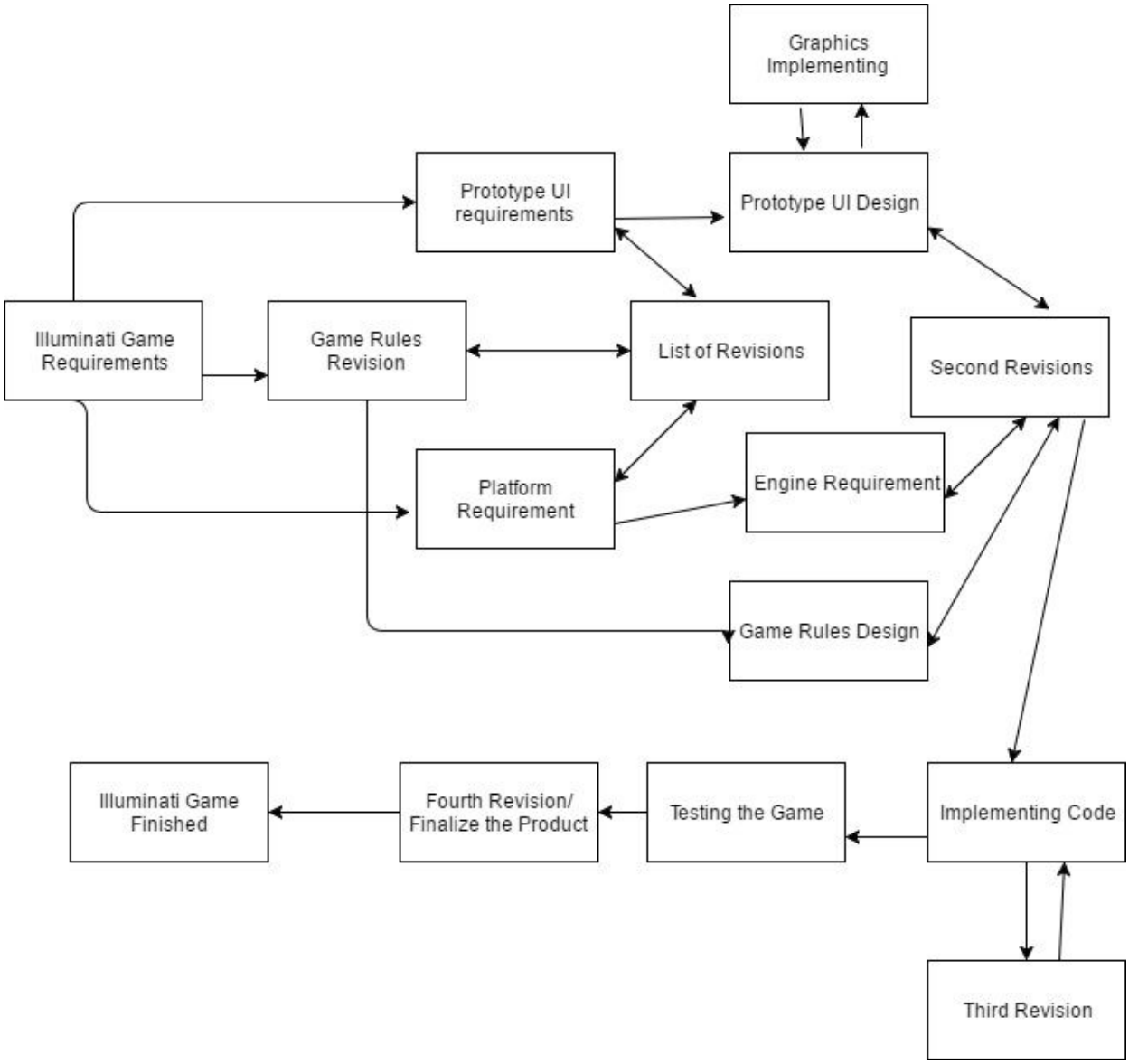
- Display: Based on the player input, the game will display their moves, and the moves of their opponent accordingly.
- Visual Studios Files (.cs): C# code is provided within .cs files for individual functions for every individual card, making each card have its own unique actions within the game for Unity3d game engine to incorporate.

Performance/Behavior Issues

Illuminati is designed to be compatible with Unity3d which requires Microsoft Windows 7 or higher. Per Unity 3d's requirements, Illuminati should also run on Mac OS X 10.8 or higher, Ubuntu 12.04 or higher and SteamOS, however it will not be tested or developed on these operating systems.

Unity3d also requires a graphics card that supports Microsoft DirectX 9 or above.

Management and Technical Constraints



Project Estimates

Historical Data Used for Estimates

A reference Function Point metric was calculated using Tiny Tools Function Point Calculator.

Reference Function Point Calculations:

Function Point Calculator FP: 237.6

Reference Estimated Person Month:

Average Function Point per Person Month: 10

Estimation Techniques Applied and Results

Estimation Technique: **Function Point**

Menu/Interface	Simple	Average	Complex
Number of User Inputs	3	2	
Number of User Outputs	5	3	
Number of User Inquiries		1	
Number of Files	2	3	1
Number of External Interfaces			

Game	Simple	Average	Complex
Number of User Inputs	5	3	
Number of User Outputs		2	
Number of User Inquiries			
Number of Files	9	7	5
Number of External Interfaces			

Estimate for: **Function Point**

Interface: 75.9
Game: 161.7
Total Function Points: 237.6
Illuminati est. Person Months: 23.76

LOC = FP*15

Illuminati est. LOC = 3,564

Estimation Technique: **Constructive Cost Model(CoCoMo)**

Illuminati will be an intermediate to semi-difficult software project.

Effort = $a(KLOC)^b$
Duration = $c(Effort)^d$

Equation Values for Effort Calculation
a = 3.2
b = 1.105

Equation Values for Duration Calculation
c = 2.5
d = 0.35

Estimate for: **Constructive Cost Model (CoCoMo)**

Effort = $3.2(3.5)^{1.105}$ **= 12.7745**
Duration = $2,5(12.7745)^{0.35}$ **= 6.0976**

Reconciled Estimate

Effort (in Function Points) Estimate:
Total Function Points: **237.6**
Effort (in CocoMo) Estimate:
Effort: **12.7745**
Time in Person Months Estimate:
Function Point: **23.76**
CoCoMo: **6.0976**
Average: **14.9288**

Project Resources

Due to the small size of our development team, each member will perform multiple jobs

Required Staff
- Lead C# programmer
- Assistant C# programmer
- Tutorial programmer

- Documentation/Librarian
- Graphic Designer
- Interface Designer
- Beta Testers

Required Hardware

- 4 Development Systems, various specifications

Due to the specifications of Unity, no special development systems are needed. All PC's will be Windows based using freely available software.

Required Software

- Microsoft Visual C#
- Microsoft Visual Studio for Unity
- Unity Personal 5

Risk Management

Project Risks

Some points of risk for this project are as follows:

- Hardware failure
- Inexperience with engine
- Engine limitations
- Hardware limitations
- Late delivery
- Changes in requirements
- Poor documentation

Risk Table

Risks	Category	Probability	Impact
Hardware Failure	Technical	20%	2
Inexperience with engine	Software	70%	1
Engine Limitations	Software	20%	2
Hardware Limitations	Technical	20%	2
Late Delivery	Organization	50%	1

Changes in requirements	Design	25%	2
Poor documentation	Organization	25%	3

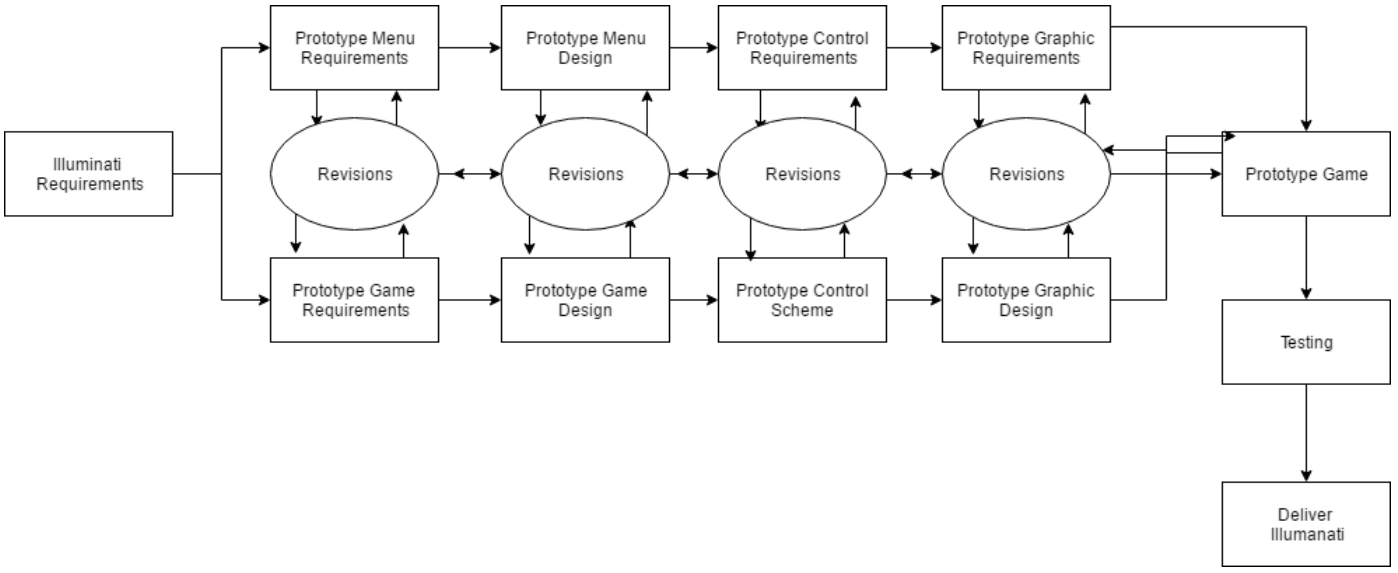
Overview of Risk Mitigation, Monitoring and Management

By identifying potential risk and proposing solutions before they occur, we can mitigate the effects should they occur. The specifications for both software and hardware will be reviewed to determine any major risk in developing the software. The potential impact of these risks on development will also be determined. These risks will then be noted, and a best course of action to deal with each will be determined. By identifying potential risk, it reduces the likelihood of more serious risks appearing. Each risk will have its own solution, to ensure that if such a thing should occur, there is a plan in place to deal with it.

Project Schedule

Project Task Set

Process Model



Framework Activities

- Customer Communication
- Planning/Design
- Risk Analysis

- Programming
- Testing
- Customer Evaluation

Task Set

- Requirement specification
- Engine construction
- Interface construction
- Help contruction
- Testing product

List of Deliverables

Documentation

Vision Document
Project Plan
Flowchart/UML Diagram
Use Cases
User Manual
Test Plan

Code

Game Prototypes
UI Mockups
Complete Game
Complete UI

Functional Decomposition

Graphical User Interface Breakdown

- Menu construction
- Card database construction

Game Engine Breakdown

- Object Handler construction
- Texture construction
- Rule implementation
- Sound Handler construction
- Input Handler construction
- Text Handler construction

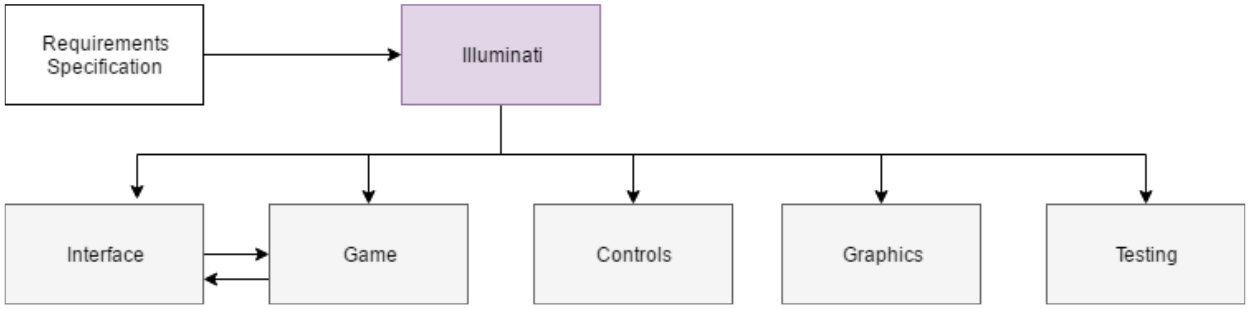
Documentation/Instruction Breakdown

- Interface help construction
- Game Rule construction
- FAQ construction
- Manual construction

Testing Breakdown

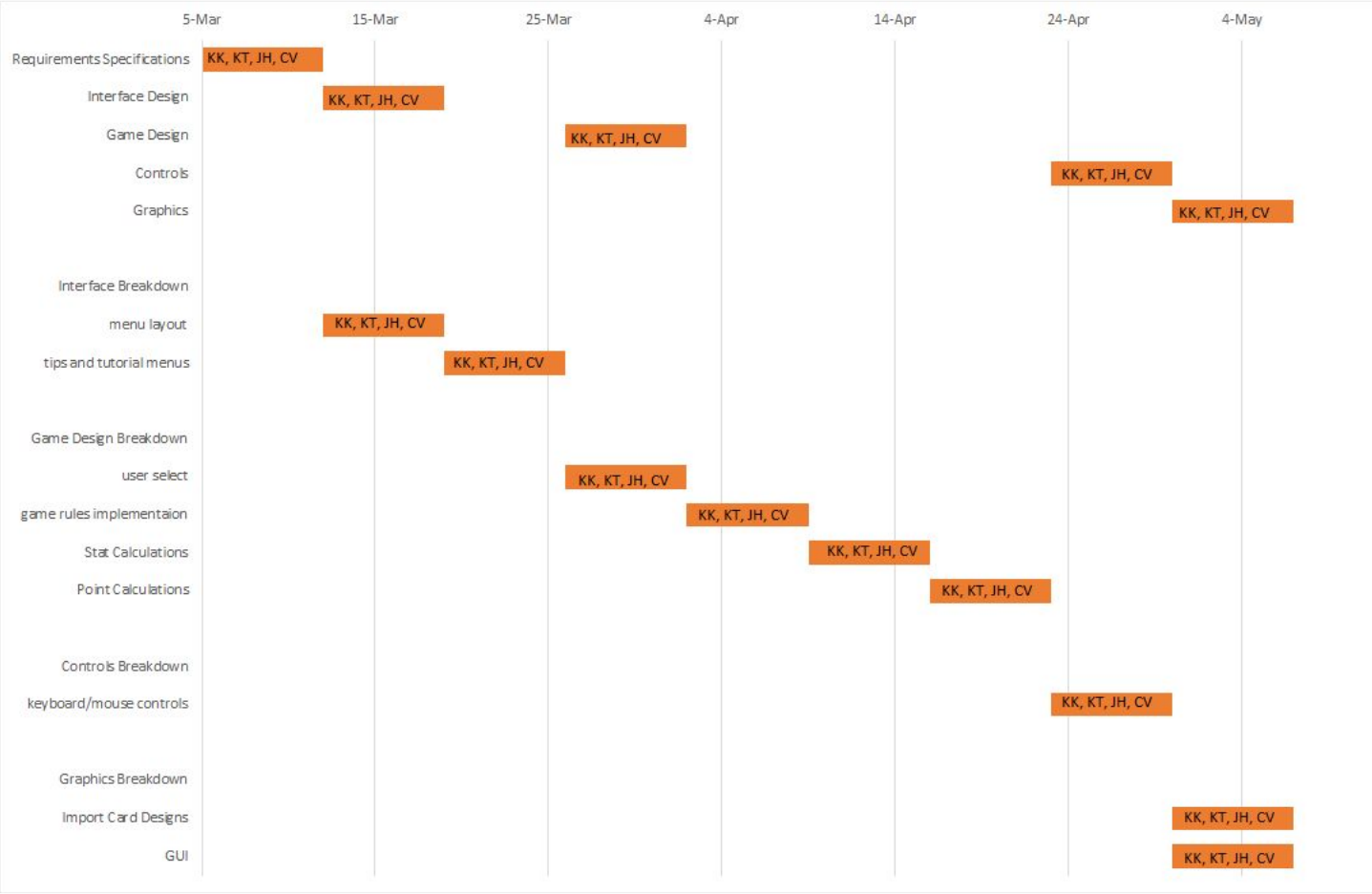
- In-house, white-box, and black-box testing

Task Network



Timeline Chart

KK = Kevin Kobata
KT = Kurt Tito
JH = Jeffrery Heng
CV = Calvin Vo



Staff Organization

Team Structure

Role Definitions

Kevin Kobata

- Game Designer: Kevin is part of the game design team.
- Game Programmer: Kevin is part of the game programming team.
- Interface Designer: Kevin is part of the interface design team.
- Help/Tutorial Programmer: Kevin is part of the Unity Help team.
- Documentation: Kevin is responsible for much of the required documentation.
- Quality Assurance: Kevin is part of the quality assurance team

Kurt Tito

- Game Designer: Kurt is part of the game design team.
- Game Programmer: Kurt is part of the game programming team.

- Interface Designer: Kurt is part of the interface design team.
- Help/Tutorial Programmer: Kurt is part of the Unity Help team.
- Documentation: Kurt is responsible for much of the required documentation.
- Quality Assurance: Kurt is part of the quality assurance team

Jeffrey Heng

- C# Programmer- Jeffrey is part of the C# programming team for scripts on individual cards and click inputs.
- Game Designer: Jeffrey is part of the game design team.
- Game Programmer: Jeffrey is part of the game programming team.
- Interface Designer: Jeffrey is part of the interface design team.
- Help/Tutorial Programmer: Jeffrey is part of the Unity Help team.
- Documentation: Jeffrey is responsible for much of the required documentation.
- Configuration Management: Jeffrey is part of the configuration management team.
- Additional Responsibilities: Jeffrey is also the primary consultant for Calvin on Unity issues.

Calvin Vo

- Game Designer: Calvin is part of the game design team.
- Game Programmer: Calvin is part of the game programming team.
- Lead Graphic Designer: Calvin is responsible for graphic design and implement graphic to the UI.
- Help/Tutorial Programmer: Calvin is part of the Unity Help team.
- Documentation: Calvin is responsible for much of the required documentation.
- Configuration Management: Calvin is part of the configuration management team.

Management Reporting and Communication

Mechanisms for Progress Reporting

A majority of progress is communicated via email. All files sent to team members will be done through GitHub or email. These communications will mainly be done informally unless special documentation is required. A log of all tests performed is kept for error tracking. Each week, a new build is created for work to occur, the old build is archived.

Mechanisms for Inter/Intra Team Communication

The development team meets at least once a week to update the team of his progress. A majority of other communications will be made in chat-logs or via email.

Tracking and Control Mechanisms

Quality Assurance and Control

Scope and intent of QA activities

The QA (Quality Assurance) team's role is to ensure that the game somewhat follows the original design specifications. If the game is deviating far from the design specifications, the QA team will notify the development team to correct and prevent future deviations. The QA team will monitor and analyze the game's quality at different stages of development. Software bugs and possible feature enhancements are also relayed to the development team.

QA Organizational Role

The QA team's organizational role is to review the game at various times during its implementation. For each major game build, the QA team will check for bugs or errors and will be identified for future builds. The QA team will directly interact with the development team, discussing errors or possible enhancements that they have been identified. In addition to identifying bugs, the QA team will ensure that the development team does not deviate too far from the initial design specifications.

Change Management and Control

Scope and intent of CM activities

The focus of the CM (Configuration Management) team is to identify and control changes to the game design as they occur. They are also there to ensure that new changes are properly implemented and reported to the interested parties. The CM will limit the effect that any changes have to the entire system. This will help minimize unnecessary changes and control necessary ones. This will minimize the amount of backtracking, lowering development time and resulting in a higher-quality product.

The CM team will oversee changes to the code or architectural design and must pass their inspection before being executed.

CM Organizational Role

The CM team will work closely with the QA team to cross-examine each document and software changes. Game designers will submit change requests directly to the CM team for inspection and approval.

The CM leader will be appointed to oversee the CM's activities. He will receive all change requests and make the final decisions regarding new changes. The CM leader also keeps a list of all submitted requests, approved or denied.

Appendix

1. Does the system require reliable backup and recovery?
   4
2. Are data communications required?
   2
3. Are there distributed processing functions?
   2
4. Is performance critical?
   4
5. Will the system run in an existing, heavily utilized operational environment?
   3
6. Does the system require on-line data entry?
   0
7. Does the online data entry require the input transaction to be built over multiple screens or operations?
   1
8. Are the master files updated online?
   3
9. Are the inputs, outputs, files or inquiries complex?
   4
10. Is the internal processing complex?
    3
11. Is the code designed to be reusable?
    3
12. Are conversion and installation included in the design?
    2
13. Is the system designed for multiple installations in different organizations?
    3
14. Is the application designed to facilitate change and ease of use by the user?
    3

CoCoMo Value Chart

	a		**b**	
mode	**basic**	**intermediate**	**basic**	**intermediate**
organic	2.4	3.2	1.05	1.05
semi-detached	3.0	3.0	1.12	1.12
embedded	3.6	2.8	1.20	1.20