Castle Quest

David Lassalle, CSE, Eric Wybenga, CSE, Devrim Dereli, EE, and Sarah Mangels, CSE

Abstract — **Castle Quest is a new take on a classic board game that utilizes electronics, software, and physical game pieces to bring players together as they venture and progress through a fantasy world. The game facilitates up to four players that will be competing against each other in pursuit of three keys scattered around the board. Each player has their own set of keys to find, hidden away in the various terrains and dungeons on the map, which will provide obstacles and conflict for the players to face. A Raspberry Pi will facilitate the game through various inputs and outputs, which include a touchscreen, multicolored LEDs, and speakers. The interactions between the player and the interface will be coded to provide an immersive setting and determine the occurrences and outcomes of encounters. Once a player has acquired all three of their keys, they must venture to the tower and defeat its three levels of combat. The first player to make it to the top of the castle wins the game.**

I. Introduction

Since the introduction of online multiplayer videogames in the early-mid 1990's, we have seen a decline in group-centric activities. Coupled with changing societal norms on the constraints placed on our youth, we see more and more interaction happening in a detached, online setting [1]. Along with the new form of interaction, we have seen the accompanying problems that go along with it like cyber bullying or impaired social development leading to awkwardness in face-to-face interaction. Our proposed solution to this problem is to bring people together in an enjoyable and interactive way via playing a board game. This will allow players to develop interpersonal skills, improve conflict resolution and learn how to be a good winner/loser. We plan to modify a board game called Dark Tower by Milton Bradley Company [6] with new electronic components and gameplay, which we will call Castle Quest.

Our approach to educating the general populace to an issue through playing games is not a unique one. In the past, people have used board games to gather individuals together and help them learn through the experience of playing together. The game *Monopoly*, or as it was originally named *The Landlord's Game* [2], was first patented in 1904 by Elizabeth Magie who originally intended it to "illustrate the economic consequences of Ricardo's Law of Economic rent and the Georgist concepts of economic privilege and land value taxation" [3]. Today though, our focus is to teach more fundamental techniques for interacting with one another; how to interact in a competitive environment, how to balance short term vs long term gain, and most importantly how to lose gracefully. In order to win Castle Quest, players must face off against each other, competing to acquire the armor, weapons, and keys to defeat the levels of the castle and win the game. They will need to determine whether to spend gold enhancing themselves or hindering their opponents as well as the route they will take on their journey. Individuals will learn to have fun while battling with friends, developing conflict resolution when the game gets tense.

Table I
Specifications

Specification	Value
Weight	<10 lbs
Battery Life	>10 hours
Height	<16 in
Width (Board Diameter)	<36 in
Age Rating	10+, not for children under 3 years

Table I shows the physical specifications of the castle and game board. Our design is proposed with the individual consumer in mind, keeping the board game light enough and compact enough to transport and having a battery life that lasts through several gameplays. Additional specifications include age rating due to the fantasy violence component of the game and the potentially dangerous nature of

D. Lassalle from Upton, MA (e-mail: dlassalle@umass.edu).
E. Wybenga from Andover, MA (e-mail: ewybenga@umass.edu).
D. Dereli from East Brunswick, NJ (e-mail: ddereli@umass.edu).
S. Mangels from Amherst, MA (e-mail: smangels@umass.edu).

electronics in general.

The remaining paper will be composed as follows; in section II the design of the game is presented, first as an overview of general background and gameplay and followed by a more detailed description of the individual blocks in the block diagram shown in Fig. B. Section III details how the project management for the construction of the game will go and what the current status of our project is. Section IV concludes the paper and lists our upcoming goals from now until the Comprehensive Design Review (CDR) and beyond.

II. Design

A. Overview

Our solution to this problem is to reinvent an electronic board game from the early 80's called *Dark Tower*. Dark tower was designed by Milton Bradley in Springfield, MA and consisted of a central tower that acted as a random event generator at each space of the game. As the player moved around the board, the tower would introduce monsters and enemies to fight with and keep track of the player's strength and game progression. Our interpretation is called *Castle Quest* and focuses on a central electronic castle in the center of the game board. The castle houses a battery pack, a Raspberry Pi, a touch screen, and our PCB. Outside of the castle, LEDs indicate which space on the board the player is in.

In a standard round, the tower will be manually rotated to one of the four players who will use the touch screen to play interact with the game and move to a new space. The LED of their chosen color will then turn off and the LED at the new location will turn on, indicating a move. Once a move is complete, the tower will be rotated the next player process repeats until a player finishes the game. At any point, the game state can be saved onto a USB and plugged into another Castle Quest game set to be continued.

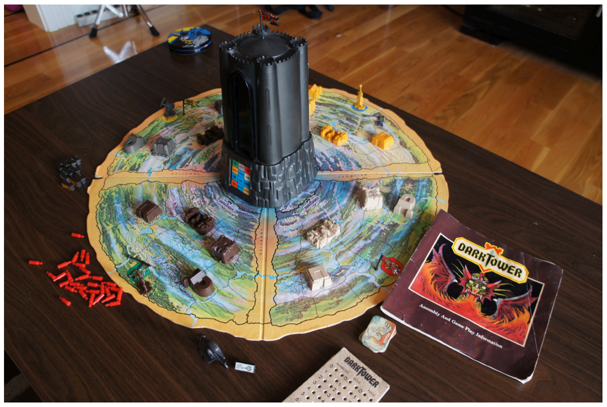

Figure A: Original Dark Tower Game

We expect this project to solve the problem we are facing by providing a new sense of excitement around tabletop games. With the electronics and portability, we believe that Castle Quest will bridge the gap between standard board games and video games in a way that is draws in the consumers for both.

Two alternatives came to mind as we discussed how to bring people together around electronic gameplay. One was the extremely popular app *Pokémon Go*, which attracted tens of millions of users in only a few weeks. The benefits of something like this are that it is quick to spread and brings people outside, but the fallbacks are that the users are still focused on their phones and that the popularity declined rapidly after the first month [4]. The second alternative was a series of games made by Jackbox Games which used the TV as a shared display for all the players who interact with the game through the internet using a smartphone or other internet-enabled device. This set up brings people together in the same room, but again has most people are focused on their phones or the TV rather than each other.

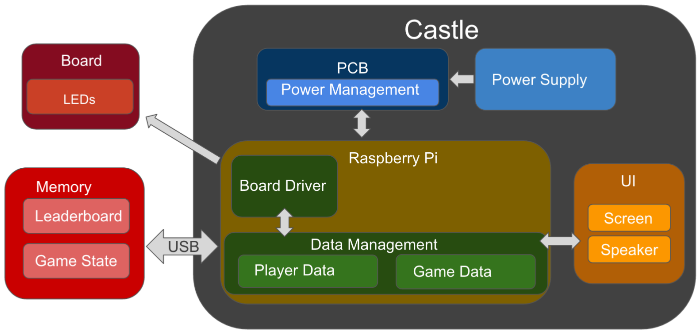

Figure B: Castle Quest Block Diagram

Most of the components of our project reside in the 3D printed castle which will need to fit our components while staying under our weight and size requirements. The Raspberry Pi in the castle is the computational powerhouse that will run all the code in the system. The Data Management block represents the the gameplay that will be run, while the UI block represents the touch screen and audio that the user will interact with. The state of game-board will also be maintained by the Pi, which will need to meet our requirement of fun and intuitive gameplay for four players. The PCB allows all of this to be possible by maintaining a steady flow of power from the battery pack into the Raspberry Pi and game board. Together these blocks create a portable, long-lasting game for up to four players to enjoy.

B. *Block 1: Castle*

The castle from the original game was very befitting to its title; a tall black tower perched on a bed of rock, standing monolithically at the center of the map. For our game, we decided to keep the placement and motion of the castle the same, but figured a different design would be more suitable for our setting.

We started off by doing a few rough sketches to come up with an overall structure of the tower, whether it should be conical, pyramidal, cylindrical,

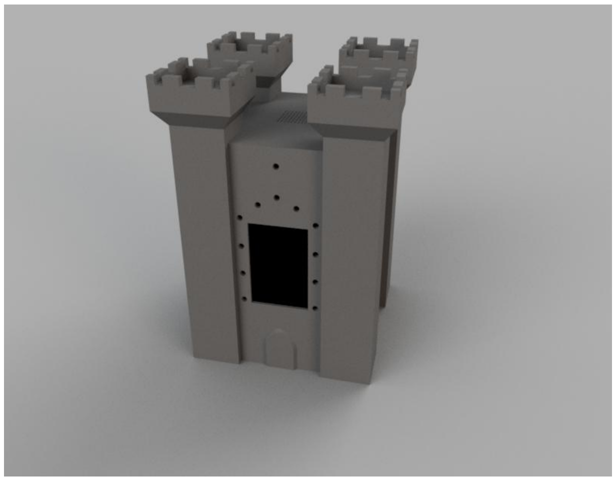
Figure C: 3D Castle Model

or rectangular. Once the structure was selected, a more detailed and dimensioned blueprint was drawn on graphing paper. The dimensions chosen were based on the size of the internal and external components that are to be placed on and within the castle, as well as few aspects of gameplay. It had to be wide enough to mount the touchscreen on the front, have enough volume to house the pi, connectors, PCB, and battery, provide a little bit of cover on the sides so that players can't see the screen during another player's turn, and still capture the relative enormity it has to the board.

Once all of that was settled, the design was modelled using a CAD software called Fusion 360, made by Autodesk [10] shown in Figure C. Currently, the tower is to be printed using the 3D printer in Marcus 5, and will most likely go through a few modifications as we continue along with the building process.

C. *Block 2: Data Management*

The Data Management will consist of a Java project running on the Raspberry Pi [5]. This code will run the game and maintain game and player state, including player inventory and stats. Data Management will essentially be the software driver for the game. The game will be developed in Java using Eclipse [7]. The first step to developing the game, which we have already completed, is to design Java interfaces to specify what classes, methods and input/outputs we will need. We have also designed a finite state machine (FSM) to govern game play which we will use to implement the game in code. The FSM specifies the inputs, outputs and actions of each player's turn in the game and can be seen in Figure D. The main method will act as the controller of the game and will loop through each player's turn according to the FSM.

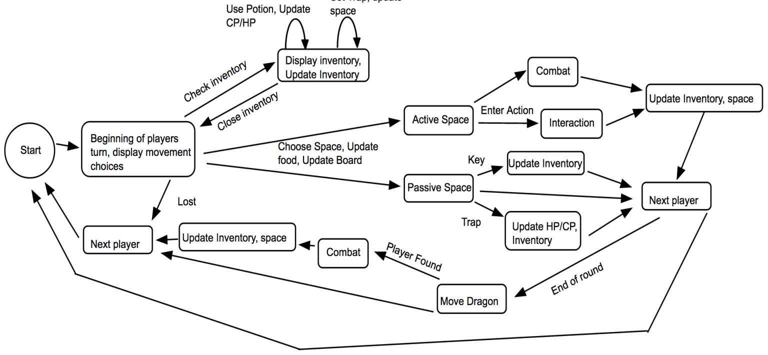

Figure D: Gameplay FSM

This block will use technical knowledge from our *Data Structures* and *Software Intensive Engineering* courses including programing and testing in java, software development lifecycle and program data structuring and storage.

In order to complete this block, we will need to review graph data structures and learn how to integrate a java project with the Raspberry Pi and touch screen.

We will test the Data Management block using JUnit testing which is a unit testing framework for Java [8]. This allows us to test common and edge cases for each method and class in the java project and to ensure that each sub-element of the code system is working properly. Next, we will functionally test the whole java project by developing game play scenarios and hand calculating the expected outputs for comparison. Finally, we will test the whole Data Management portion of the system when the entire project has been integrated simply by playing the game and verifying the proper executing using the FSM and gameplay rulebook we have developed

D. Block 3: Game Board

The game board is what provides the players with spatial awareness of what is happening in the game. As such, our board must clearly display each player distinctly in a way that they will be able to see at a glance how they should progress through the board.

The physical board itself needs to be lightweight enough to be feasibly transportable while being strong enough to survive repeated transportation. The board will be constructed from a layering of cardboard, medium density fiberboard (MDF), and waterproof glossy poster paper. The bottom layer will be constructed from cardboard and the soft flexible nature of the material will be used to inlay the wires running from one addressable LED to the next, representing the players as they move around the board. This will protect the wires from being brushed up against, potentially disconnecting a wire, and will absorb some impact from a dropped board. The middle layer is the MDF and provides structure to the game board. MDF is a dense, sturdy, and smooth material. Since it lacks a wood grain, it is easily cut, allowing us to cut holes for each of the LEDs to show through. The drawbacks to MDF are that its density makes it a very heavy material which we plan on overcoming by using a thinner sheet (~1/8”) as well as its tendency to soak up water which will be covered by an oil finish as well as the top layer, the glossy poster paper. The poster paper will protect the lower layers from spills on the board and will provide a surface to design the graphical game board. Holes will be cut in the poster to allow the LEDs to shine through and the LED holes will be waterproofed with hot glue, protecting the LEDs and lower layers as well as diffusing the LED light. The layers will be attached to one another via a glue adhesive.

The player processing will occur in the CPU of the Raspberry Pi and the information will be sent out to the LEDs via the Pi’s I2C GPIO pins. Techniques for hardware interface learned from *ECE 353/354 - Computer Systems Lab 1 and 2* will be employed to interface to hardware as well as to write player positions to external memory.

To test this block, shorter chains of addressable LEDs will be strung together on a breadboard and the LEDs will be tested to see that they can accurately be selected to show a specific color. We then will test the capability for multiple entities to coexist on one point by alternating the color of the node between the colors associated with all current residents of that spot. Finally, the chain of LEDs will be expanded to encompass all board spaces and will be attached to the physical board.

By performing the above tests, we verify that the chain of LEDs has no malfunctioning units and that all edge cases are covered for the physical hardware display.

E. Block 4: User Interface

In order to have a fun and intuitive game, the user interface needs to provide a sleek and simple means to interact with the game driver. The UI will be a graphical user interface that receives and displays data to and from the user via a touch screen and speaker in the castle. It will be responsible for displaying the status of the game for each player and relay gameplay decisions to the data management block to invoke game progression and game board updates.

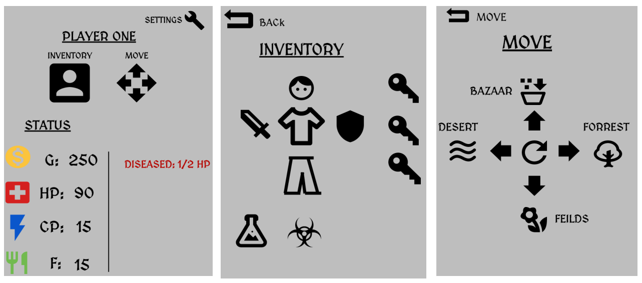

Figure E: Current UI Mock-ups

The Raspberry Pi has been configured with a 3.5mm headphone jack and vertically-oriented resistive touch screen. During fight sequences, tower rotation, and other game events, short audio clips will be played through a speaker on the top of the castle from the 3.5mm headphone jack. Meanwhile, the Java Swing GUI will be displayed on the touch screen for user interaction [9].

In *CS 121 – Introduction to Problem Solving using Java*, we briefly learned about Java's native GUI capabilities and designed a rudimentary sliding text screen from user input. Using Eclipse with the WindowBuilder plug-in, we will learn more complex operations that the Java client can provide and how to create and display animations.

Once the GUI is complete, a transition though of each UI screen will be done to ensure that the interface works correctly. For a full test, the UI block and data management block will need to be tested together with a play-though of the game. Depending on the results of the play-through, there will likely be many tweaks that need to be made to ensure the UI and data management communicate correctly.

F. Block 5: PCB

Initially we had decided to utilize the PCB for I/O expansion due to the large number of LEDs that need to be utilized by the board, but after receiving some feedback and advice from our MDR evaluation, we decided to redirect the purpose of the PCB towards power management. A variable voltage regulator will be built on the PCB in order to provide linear power to the electronics.

A rather robust and simple design was found online through the DIY site Instructables.com [2]. The design implements an LM317 voltage regulator IC, a 1N4001 diode, and several linear circuit components. The current design can source between 7 to 14V and supply 1.6 to 6V, which is well within the range of the operating voltage needed to power the pi. Techniques learned in the project portion of *ECE 324 – Electronics II*, such as datasheet analysis, implementation of basic circuit theory, and spatial allocation for hardware will come in handy for building and debugging.

Currently the circuit is being built on a bread board and being tested for functionality. Once, it performs to spec, the components will be soldered onto a protoboard to see how the final PCB will fit and function with the other components. The final step will be to build a custom PCB for the circuit using a CAD program. We plan on learning and using Eagle CAD to make our final design.

III. Project Management

Prototype Castle Model	Complete
Prototype Game Board and UI	Complete
Finalize Gameplay	Complete
Code System Overview	Complete
PCB Design – Breadboard Mock Up	Incomplete

Figure F: Status of MDR Goals

So far, the full Castle Quest game has been designed, including a layout of the game board, the CAD model for the 3D printed castle, graphical designs for the UI and a game play design which includes a finite state machine, java interfaces and a well defined main code outline. As a team we thought that we met our MDR deliverables by the time of our presentation however our evaluators felt our deliverables were not well defined. Our evaluators felt we should have made more progress by that time. In the future we plan to define our goals and deliverables more clearly and communicate with our evaluators so everyone is on the same page. We now need to finish implementing the java code, design and print the PCB, execute the code for the LED's and UI, 3D print the castle and build the board. A Gantt chart demonstrating the current progress and future plan is included in Figure G.

	Sept-Nov	MDR	Dec	Jan	Feb	CDR	Mar	Apr	FPR
Design Castle Model and Board Layout	x								
Design Game Play	x								
Design UI	x								
Print and Assemble Castle			x	x					
Implement UI			x	x					
Implement LED Code			x	x					
Implement Gameplay			x	x	x				
Prototype Full PCB				x	x				
Construct Board				x	x				
Integrate Code Systems					x				
Print PCB					x				
Final Testing and Debugging							x	x	
Final Game Assembly							x	x	

Figure G: Gantt Chart

Each week our team meets for about an hour or two on Wednesday evenings to discuss progress and questions and to make sure we are on schedule. We then meet with our advisor on Friday afternoons for a half hour meeting to check in. We often use this time with our advisor to practice presentations or review progress on key goals. We work throughout the week in the SDP lab both individually and as a group. Each week, David sends an email with the goals and task of the week broken down for each team member and we plan to have those accomplished by the Friday meeting with our advisor. Less formal communication in our group is done through a group Facebook chat and in person as we spend a lot of time together. We have split the responsibilities of the project down into individual and smaller groups as follows.

Game implementation in Java - Sarah, David
PCB design, power management - Devrim
Game board design and implementation - Eric
LED layout and programming - Eric, Devrim
UI implementation in Java - David, Sarah
Castle design and 3D printing – David

We think these tasks fit well with each team member's areas of interest and experience. Devrim, being the sole Electrical Engineer of the group, has taken on most of the hardware based project tasks. Eric (CSE) has previous experience using Raspberry Pi and a strong interest in the LED programming. David (CSE) has experience using CAD software and 3D printing for previous projects and is interested in Java based UI design. Finally, Sarah (CSE) has a strong background with Java programming and is interested in working on the integration of all components of the project.

Ultimately, as this is a team project, we are all expecting and willing to help each other out. Specifically, so far, David and Sarah have worked closely on designing the Java interfaces to implement the game in Java and Devrim and Eric have been working together to design the use of the LED's and the game board. We anticipate a lot of collaboration when we begin integrating the various components of the project.

IV. Conclusion

As of now, Team 13 is slightly behind our projections from the beginning of the project. We met most of our MDR deliverables and have designs for each subsystem of our block diagram, but our MDR deliverables were too weak, meaning that there is a larger portion of work to be done for the spring. Our project work process consists of weekly deliverables for each team member and we will continue to do this throughout the spring semester.

Between now and CDR, we will be generating code from our designs and finalizing each subsystem. Once that is complete, our goal is to integrate most, if not all, of the systems for our CDR evaluators to review.

As for any project, we expect a significant amount of debugging for the data management, board driver, and UI codes as well as hardware debugging for the PCB. One of the larger difficulties will be printing the tower, due to a lack of organization around the printer in M5.

Post-CDR, the main goal will be full integration for all systems and aesthetic finalization of the castle and game board.

References

[1] Herring, Susan C. "Slouching toward the ordinary: Current trends in computer-mediated communication." New media & society 6.1 (2004): 26-36.
[2] Pilon, Mary. "Monopoly's Inventor: The Progressive Who Didn't Pass 'Go'." The New York Times. The New York Times, 14 Feb. 2015. Web. 18 Dec. 2016.
[3] Parlett, David (1999). The Oxford History of Board Games. Oxford University Press. p. 352. ISBN 0-19-212998-8.
[4] Kawa, Luke, and Lily Katz. "These Charts Show That Pokemon Go Is Already in Decline." Bloomberg.com. Bloomberg, 22 Aug. 2016. Web. 19 Dec. 2016.
[5] Raspberry Pi Ltd. (2016, October). [Online]. Available: https://www.raspberrypi.org/documentation/hardware/computemodule/RPI-CM-DATASHEET-V1_0.pdf.
[6] "Dark Tower (game)", En.wikipedia.org, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Dark_Tower_(game). [Accessed: 23- Jan-2017].
[7] *Eclipse*. The Eclipse Foundation, 2017. [Online]. Available: https://eclipse.org/. [Accessed: 23-Jan-2017].

[8] JUnit, 2017. [Online]. Available: http://junit.org/junit4/. [Accessed: 23-Jan-2017].

[9] Oracle, “A Swing Architecture Overview” [Online]. Available: http://www.oracle.com/technetwork/java/architecture-142923.html. [Accessed 23-Jan-17].

[10] Fusion 360. Autodesk, 2013. Available: http://www.autodesk.com/products/fusion-360/overview

FIGURES

[A] Tested.com, 2016. [Online]. Available: http://d2rormqr1qwzpz.cloudfront.net/uploads/0/5/35941-darktower_field.jpg. [Accessed: 11 - Dec - 2016]