

# 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 placed around the board. Each player has their own set of keys to find, hidden away in the kingdoms on the map, which will be obstructed by conflict and danger for the players to face. A Raspberry Pi linked to an Arduino Nano through an I2C connection facilitates the game through various inputs and outputs, which include a touchscreen, multicolored LEDs, and a speaker. The interactions between the player and the interface is 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 attempted to address this issue by modifying a board game called Dark Tower by Milton Bradley Company [6] with new electronic components and gameplay, which we have called Castle Quest.

Our approach to educating the general populace to an issue with 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 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	>4 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 was 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 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

## Team 13 Castle Quest

the game will go and what the current status of our project is. Section IV concludes the paper and lists a summary of our milestones (CDR, FPR, Demo Day, etc.) as well as a synopsis of our next steps for the project.

## 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, an Arduino Nano, and our PCB. Outside of the castle, LEDs indicate which space on the board the player is in and a speaker plays audio feedback to the player.

In a standard round, the tower will be manually rotated to one of the four players who will use the touchscreen to 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 where the process repeats until a player wins 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.

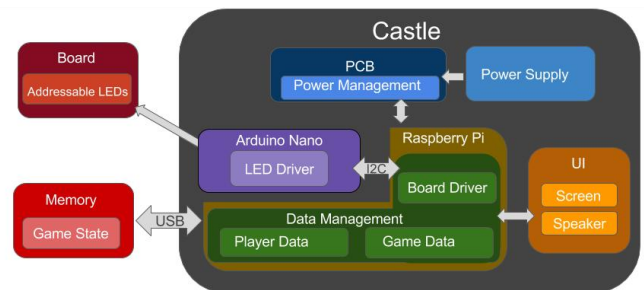


Figure B: Castle Quest Block Diagram

Most of the components of our project reside in the castle which fits our components while staying under our weight and size requirements. The Raspberry Pi in the castle is the computational powerhouse that runs all the code in the system, while the Arduino Nano receives move commands from the Pi and updates the LEDs on the board. 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

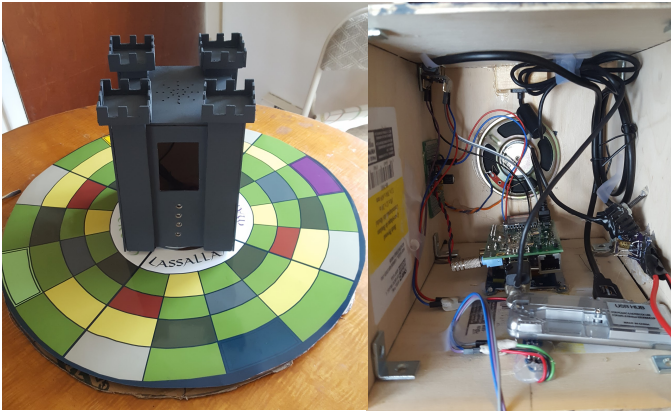
## Team 13 Castle Quest

will interact with. The state of the game-board will also be maintained by the Pi, which meets 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, or rectangular. Once the structure was selected, a



(a) (b)  
Figure C: Exterior (a) and interior (b) of the castle.

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 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]. The initial project called for 3D printing the tower based on the CAD model, however, due to functional setbacks of the printers in the M5 makerspace, the castle was instead constructed using craft wood, wood adhesive, and magnets to allow the front panel of the castle to be removed so that the internals could be accessed during the assembly process. Figure C(a) shows the final external build of the castle, and Figure C(b) shows the inside of the castle with all mounted subsystems. Once the tower was completely assembled, the rotating base was fitted to the base of the castle.

### C. Block 2: Data Management

The Data Management will consist of a Java project running on the Raspberry Pi [5]. This code runs the game and maintain game and player state, including player inventory and stats. Data Management is essentially the software driver for the game. The game was developed in Java using Eclipse [7]. First, we designed Java interfaces to specify what classes, methods and input/outputs we needed. We also designed a finite state machine (FSM) to govern game play which was 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 acts as the controller of the game and loops through each player's turn according to the FSM.

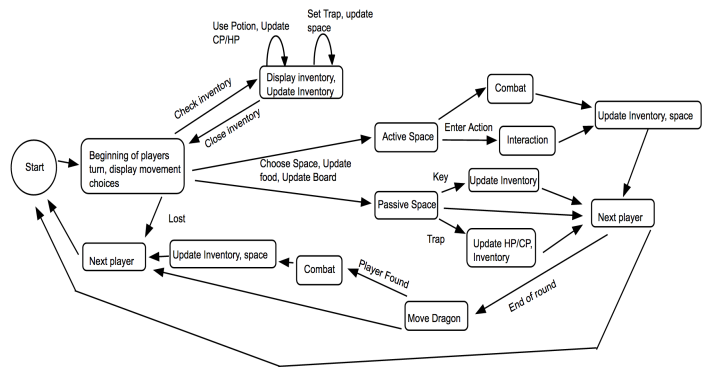


Figure D: Gameplay FSM

This block used technical knowledge from our *Data Structures* and *Software Intensive Engineering*

## Team 13 Castle Quest

courses including programming and testing in java, software development lifecycle and program data structuring and storage.

We were able to test the software we developed by running the software on our computer and playing the game through the eclipse command line. This way we were testing the software independently from the hardware and other components to ensure that each sub-element of the code system was working properly. Next, we functionally tested the whole java project by developing game play scenarios and hand calculating the expected outputs for comparison. Finally, we tested the whole Data Management portion of the system when the entire project was integrated simply by playing the game and verifying the proper executing using the FSM and gameplay rulebook we 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 clearly displays 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 is constructed from a layering of cardboard, medium density fiberboard (MDF), and waterproof glossy poster paper. The bottom layer is constructed from cardboard and the soft flexible nature of the material is used to inlay the wires running from one addressable LED to the next, representing the players as they move around the board. This layer also serves to protect the wires from being brushed up against, potentially disconnecting a wire, creating a dangerous scenario, 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, which allowed us to cut holes for each of the LEDs to show through. The drawbacks to MDF is that its density makes it a very heavy material, which we have overcoming by using a thinner sheet (~1/8"), as well as its tendency to soak up water.

Finally, we have the top layer, the glossy poster paper. The poster paper serves to protect the lower layers from spills on the board and provides a surface for the graphical game board. Originally, we were planning to cut holes in the poster paper to allow the light to shine through but this step proved to be unnecessary in our end result. The LED brightness and the thinness of the poster paper allowed the colors to shine through each of the colored spaces and to be clearly differentiable from one another without the need to puncture the graphic. This also proved to be beneficial in the game's storage because the board itself looks cleaner when the game is turned off. The LEDs on the tower itself still needed waterproofing and our solution to this was to fill the holes with hot glue, protecting the LEDs. This served to protect the internal components as well as to diffuse the LED light. The layers are attached to one another via a glue adhesive.

The player processing occurs in the CPU of the Raspberry Pi and the information is sent out to the player controller located in the Arduino Nano. The controller is instructed via the Pi's I2C GPIO pins. Techniques for hardware interface learned from *ECE 353/354 - Computer Systems Lab 1 and 2* is employed to interface to hardware as well as to write player positions to external memory.

To test this block, shorter chains of addressable LEDs were strung together on a breadboard and the LEDs were tested to see that they can accurately be selected to show a specific color through mediums such as hot glue and poster paper. We also tested 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. The chain of LEDs was expanded to encompass all board spaces and was attached to the physical board.

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

### *E. Block 4: User Interface*

In order to have a fun and intuitive game, the user interface provides a sleek and simple means to



## Team 13 Castle Quest

interact with the game driver. The UI is a graphical user interface that receives and displays data to and from the user via a touch screen and speaker in the castle. It is responsible for displaying the status of the game for each player and relaying gameplay decisions to the data management block to invoke game progression and game board updates.

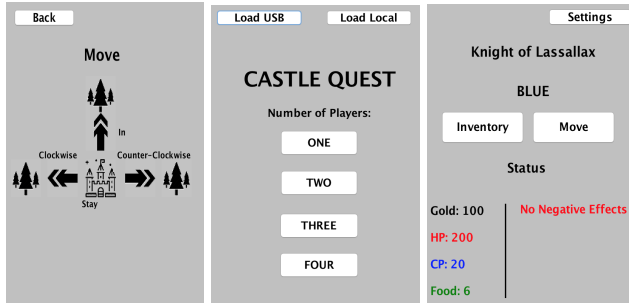


Figure E: GUI Screenshots

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 are played through a speaker on the top of the castle from the 3.5mm headphone jack to a small amp then to a small speaker driver. Meanwhile, the Java Swing GUI is 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 created and learnt about more complex operations that the Java client can provide and how to display animations.

Java Swing ended up being a very intuitive system for designing interfaces, which made the visual component of the code much simpler to create, although the background processing for maintaining the game state and player stats was more complex than anticipated.

Once the GUI was complete, a transition through each of the UI screens was done to ensure that the interface works correctly. For a full test, the UI block and data management block had to be tested together with a play-through of the game. Each play-through used different variations of number of

players, as well as who the players were played by. The results of the play-throughs highlighted many tweaks that needed to be made to ensure the UI and data management blocks communicated correctly. The first few tests found bugs such as player stats not being updated correctly and situations where certain spaces couldn’t be accessed. After these were fixed, the next tests were used to adjust the levels of difficulty in the game to make the experience more fun.

### 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 is the main IC utilized 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 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 came in handy for building and debugging.

The design was implemented through Eagle CAD[11], and the gerber files for the board were sent to OSH Park[12] to be printed. A total of three boards were printed, one of which was used as a testing board, another as the final PCB, and one was left as a spare. The final board drew power from a 7.4v, 4400mAh lithium ion battery. With that battery, the PCB was able to source 5.4v to the Pi for approximately three hours.

## III. PROJECT MANAGEMENT

Our team was able to complete all our FPR deliverables listed in Table II by our FPR presentation on April 14<sup>th</sup>, 2017.

Table II

## Team 13 Castle Quest

FPR Deliverables	
Deliverable	Date Completed
Final Game and Board Assembly	4/4/17
Software Testing	3/27/17
Hardware Testing	4/6/17
Integration Testing	4/12/17
Tower and Board Aesthetics	4/10/17

Each week our team met for about an hour or two on Wednesday evenings to discuss progress and questions and to make sure we were on schedule. We then met with our advisor on Friday afternoons for a half hour meeting to check in. We often used this time with our advisor to practice presentations or review progress on key goals. We worked throughout the week in the SDP lab both individually and as a group. Each week, David sent an email with the goals and task of the week broken down for each team member and we planed to have those accomplished by the Friday meeting with our advisor. Less formal communication in our group was done through a group Facebook chat and in person as we spent a lot of time together. We were able to split the responsibilities of the project down into individual and smaller groups when necessary, and came together towards the end to integrate our components. In the final week before FPR we all got together to play the whole game through a few times.

Each team member was able to incorporate and utilize their areas of interest and experience. Devrim, being the sole Electrical Engineer of the group, took on most of the hardware based project tasks. Eric (CSE) has previous experience using Raspberry Pi and worked on the LED programming. David (CSE) has experience using CAD software and is interested in Java based programming so he developed the Java game and designed the CAD model for the tower. Finally, Sarah (CSE) has a strong background with Java programming and is artistically inclined so she worked on building the tower and designing the game graphic and well as integration of the project.

Ultimately, as this was a team project, we were all expecting and willing to help each other out. Specifically, David and Sarah worked closely on designing the Java interfaces to implement the game

in Java and Devrim and Eric worked together to design the use of the LED's and the game board. We all collaborated when we began integrating the various components of the project.

#### IV. CONCLUSION

Our MDR report's conclusion outlined that Team 13 was behind schedule due to the weak targets set for the Fall 2016 semester, as indicated by our evaluators Professors Irwin and Soules. However, between MDR and CDR, the initial gameplay, game board, and LED communication were operational, with our power management PCB implemented, but not integrated into the full system.

At FPR, all the subsystems were implemented and integrated into the tower, with the battery providing power to the PCB which stepped-down the voltage for the Pi and remaining components to run off of. The result was a fully-functional digital board game with saving capabilities that hit a majority of our specifications set in PDR, and received praise from viewers at the SDP17 Demo days.

In summary, the completed Castle Quest project provides a digital board game experience with lights and sound for up to four players around a table. At an average of 1.5 hours, each game played can be saved at will to the local machine or USB drive. All of this was accomplished on time and under budget for our projections and deadlines from PDR.

Looking forward, Eric Wybenga and David Lassalle are working with Robert Jeffways of Robert Jeffways Jr. LLC to examine possibilities of improvements to the system, as well as licensing and patents related to Castle Quest and its subsidiary code.

#### 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/R>

## Team 13 Castle Quest

- PI-CM-DATASHEET-V1\_0.pdf.
- [6] "Dark Tower (game)", En.wikipedia.org, 2017. [Online]. Available: [https://en.wikipedia.org/wiki/Dark\\_Tower\\_\(game\)](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>
  - [11] Eagle CAD. Autodesk, 2015. Available: <https://www.autodesk.com/products/eagle/overview?mktvar002=695723&gclid=CPKPttzT1tMCFQSHswodFOMERw>
  - [12] OSH Park. 2016. Available: <https://oshpark.com/>

## FIGURES

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