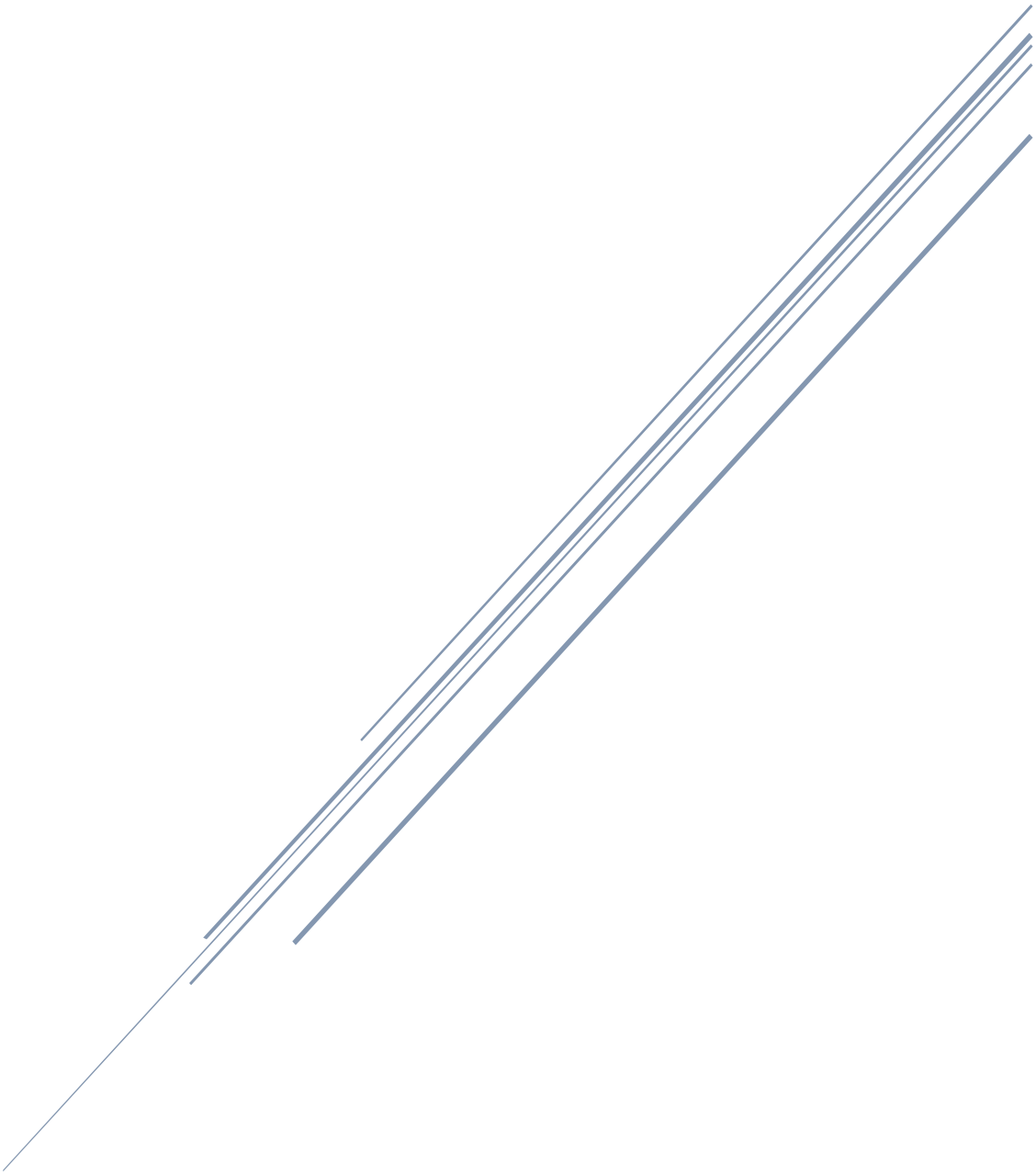
DESIGN AND DEVELOP A PIECE OF SOFTWARE TO PLAY AN ORIGINAL COLLECTIBLE CARD GAME

Level 3 Extended Project (7993)



Candidate Number: 6029
Centre Number: 74303

Candidate Name: Alex Goodall
Centre Name: The British School of Brussels

I knew when choosing this project, it would require a lot of time and effort. However, I saw this as the perfect opportunity to further develop my Object Orientated Programming (OOP) skills above the curriculum. OOP is a programming paradigm the uses the idea of objects, instantiated by structured classes (bodies of code) that store data and methods to manipulate their data. My dream job is to be a software engineer because I thoroughly enjoy problem solving and creating things, but I was not sure how I could turn this idea into a large-scale project. However, along with my own personal interest in card games, I decided that designing a new Collectible Card Game (CCG) and developing a piece of software to play it would be the perfect project for developing my OOP skills and understanding, while also being very enjoyable. A CCG is type of card game that consists of playing with numerous uniquely designed cards in a particular format of game. My interest in them developed by playing a wide variety of CCGs as I grew up and even now. After fully reviewing my options for my title, I decided on 'Design and develop a piece of software to play an original collectible card game'. I decided to work in Visual Studio.NET because this was the compiler I was most used to working in. I also worked using the GUI class library, windows forms, because this was the environment I planned to work in for my A-Level Computer Science project, so naturally I wanted more experience working with windows forms.

There are a wide variety of online CCGs with different rulesets, formats and mechanics (how the game works) so to design my game I set out to investigate popular online CCGs for inspiration and ideas. Because I was not fully accustomed with working in windows forms I also set out to research into configuring windows forms controls *(Buttons, Labels, Pictureboxes*, etc.). When I started my project, I had some exposure to OOP but it was limited and I still had a lot to understand and learn. So, I set out to research the core ideas of OOP; how to write relevant, flexible and encapsulated classes. Making my code flexible would allow me to easily change the direction of the design of my classes during the development of my software, rather than re-writing large bulks of code. I was convinced the challenging part of this project was sourcing and designing my own set of original cards as I was not very experienced with graphical design. So, when I devised my Gantt chart at the beginning of the project I allocated a lot of time to the card development. I had decided at this point I would not host a game server, which meant 2 real players would not be able to play against each other over the internet. Hosting a game server would overcomplicate my project and require a monetary investment. So, my intentions were to produce a piece of software to play my CCG against an Artificial Intelligence (AI). This also made me set out to research into AI development for games. My desired outcome was to produce a standard CCG with unique mechanics, aesthetic graphics, enjoyable to play and straight-forward to learn. I planned to assess these goals at the end of my project by testing the game myself and with other people to get their feedback. My initial course of action is described below in *Figure 1.0*, the red and blue are simply to differentiate between the different rows because there is a lot of overlap.

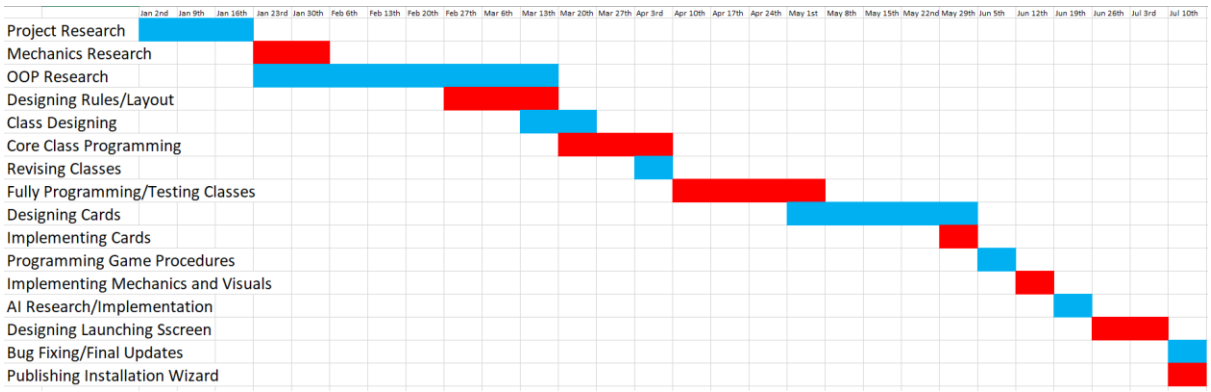


Figure 1.0 - Gantt chart

My initial research was focused on investigating popular CCGs. My own personal experience in playing multiple different online and real-life CCGs helped me thoroughly evaluate an array of card games, referring primarily to their enjoyability, mechanics (how the game works) and rulesets, including how easily these could be developed. I wrote notes on aspects of the game I liked, aspects I did not like and aspects I could consider including in my game. I found a lot of the games I evaluated operated in a very similar format, so I decided I would try to stay away from the traditional style of CCG, where the idea is to kill your opponent by playing cards that damage your opponent. However, I was still able to take away things from games like this such as: aspects that made them enjoyable and mechanics that I found useful or interesting. The 4 very different games that impacted my decision making when designing my game were *Hearthstone, Magic 2015, Gwent* and *Chronicle: Runescape Legends*. Firstly, *Chronicle: Runescape Legends* was the most graphically developed because it used 3D character models, along with some amazing animations and effects. *Figure 2.0* illustrates the extent of *Chronicle's* graphical development. It was also very different from your traditional card game with a unique format and ruleset. "From inside a magical book, players use their deck of cards to create a quest for their legend. Legends earn gold as they progress through the quest, fighting enemy cards, which players can spend to aid their legends. Enemy players will be creating their own quest and may steal gold or weapons. At the end, players do battle against each other, with the victor being the last one standing."[1] This was very unique because you never really interact with your opponent until the end of the game. I also noticed that this game took ideas (for example a weapon card) from more traditional CCGs and incorporated these ideas with its own unique format. This highlighted the idea I wanted to employ when designing my game. However, with the graphics being developed far beyond my scope, the scope of Visual Studio.NET and the gameplay being very different, I found it hard to take anything more away from this game.

Figure 2.0 - Chronicle: Runescape Legends gameplay

Hearthstone and *Magic 2015* were both very similar traditional CCGs (as described earlier) with a very similar format and ruleset behind them. Both *Hearthstone* and *Magic 2015* consist of this format. "In matches, players use their card deck, representing summonable minions, spells, and other actions, along with their selected Hero's unique power, to try to defeat their opponent by reducing their health to zero before the opponent can do the same to them."[2] The main thing that separated these games was enjoyment; *Hearthstone* I found was far more enjoyable because it had a stronger element of luck. Effective use of luck is arguably the most important thing for not only a card game but most online games, because in my opinion it is what makes a game enjoyable along with adding an element of replay-ability (desire to play the game again). I heavily took this into consideration when later designing my cards and developing my game, using the element of luck effectively to make my game enjoyable and interesting. I also used the idea of the end turn button (a simple button that ends your turn when you click it) and the hovering mechanic (hovering over a card results in a larger display of the card so you can read it more easily) from *Hearthstone* and implemented it in my game. I also took away the idea that there is a limit of 10 cards you can hold in

[1] "Chronicle: Runescape Legends Wiki". *Chronicle: Runescape Legends Wiki.* Last modified 2017. Accessed January 7, 2017. https://chronicle.gamepedia.com/Chronicle:_RuneScape_Legends_Wiki.

[2] Jeffrey Matulef. "Hearthstone - The Collectible Card Game That Could Convert You". *Eurogamer.Net*. Last modified 2017. Accessed January 8, 2017. http://www.eurogamer.net/articles/2013-03-26-hearthstone-the-collectible-card-game-that-could-convert-you.

your hand. I used this idea because this meant I could have a static amount of *PictureBoxes* in the windows forms that displayed the cards. Having to programmatically add and remove *PictureBoxes* for each card, would have been more complicated to program in Visual Studio.NET and could have resulted in flashing and glitching, meaning it would not run smoothly.

Finally, *Gwent* is a very new card game (public beta released in 2017) with different, quite simplistic yet unique gameplay. *"Gwent* is a turn-based card game between two players, each player must play one card each turn from a deck of twenty-five to forty cards with a faction that offers different play styles and unique abilities."[3] The reason Gwent is fast and enjoyable is because you play 1 card per turn and you only start with 8 cards, so the idea is to deceive your opponent, capitalise on small advantages and plan out your strategic play, very like Poker and its variations. Gwent is fairly simplistic because it used to be the in-game card game for the Role-playing Game (RPG) *The Witcher 3: Wild Hunt*. However, this simplistic gameplay makes it unique and easy for new players to pick up and has helped it become a successful stand-alone game. This game diversifies its strategy and gameplay by including different factions that have different play-styles. I used this idea and decided to incorporate 4 different characters to play with different abilities and cards associated with them. I believe the format of *Gwent* will be seen in a new branch of CCGs that will become very popular simply because of its quick, simplistic and enjoyable gameplay. For these reasons, I decided to use elements of *Gwent's* layout (*Figure 2.1*) along with its format and ruleset to shape my game. Using my findings, I could devise the rules for my game and decide which mechanics and visual effects I would try to develop for my game (See Appendix A for the detailed set of rules used in my game). Along with *Gwent's* basic rules I decided to add some of my own mechanics; 4 champions to choose from with unique abilities, character cards with 3 different identifiers (species, type and attack power), 3 distinct types of spell cards (aura, buff and instant). These ideas set me apart from simply re-creating *Gwent* and my project from now on was aimed at people like me, who enjoy playing several varieties of CCGs and mastering their gameplay.

Figure 3.0 - Layout of Gwent, The Witcher Card game

My next step was to research development methods. I figured the ideal way to approach this was to develop a pilot software. This would allow me to experiment with visual effects, different windows forms controls and ways of designing my classes. Any problems I came across I could research into using websites such as 'stackoverflow.com'.[4] I developed a simple piece of software using a very standard layout; similar to all the card games I researched into, apart from *Chronicle: Runescape Legends* (See Appendix B for notes on the elements I included). My pilot software simply dealt cards and displayed them. I used the windows forms *PictureBox* control to show the image of the cards, because this was the recommended way to display high resolution images. I then needed to find a way of handling all 40 of the *PictureBox DoubleClick*, *MouseEnter* and *MouseLeave* events, to trigger an event when the displayed cards are interacted with (clicked or hovered-over). So, I firstly considered effective ways to manage groups of objects (*PictureBoxes* in this case) and decided to use the Visual Basic.NET (VB.NET) *Collection* class, which is used for managing and grouping related

[3] "Gwent: The Witcher Card Game". *En.Wikipedia.Org*. Last modified 2017. Accessed January 13, 2017. https://en.wikipedia.org/wiki/Gwent:_The_Witcher_Card_Game.

[4] "Stack Overflow - Where Developers Learn, Share, & Build Careers". *Stackoverflow.Com*. Last modified 2017. Accessed January 18, 2017. https://stackoverflow.com/.

objects.[5] I used the *Collection* class because it was a dynamic list (changes size automatically when removing or adding objects) with the ability to remove an object by index or reference (instance). Adding was also not a problem - with the *Collection.Add* procedure the object was added to the end of the list. The *Collection* class is also very flexible so I used it for grouping and managing many different related objects throughout my project. I added all the *PictureBoxes* to a collection. I then used an iteration loop to access each of the *PictureBoxes,* and used *addhandler* to address their events to the corresponding procedures I had coded for handling the *PictureBoxes'* events. This was repetitive but necessary and also much better than handling all 40 of the *PictureBox's* events in different procedures which would have resulted in 120 procedures and a lot of code. However, after reading an article on *addhandler* I realised there was a better way to do this.[6] I wrote a class (static body of code that stores data) called *CardHolder* which stores a *Card* and a *PictureBox* in its members. The constructor (method that instantiates a class) of *CardHolder* had a *PictureBox* control as its only parameter, so when the class was instantiated I used *addhandler* within the constructor to address the *PictureBox DoubleClick*, *MouseEnter* and *MouseLeave* events to the corresponding procedures.

Furthermore, I would have to code a lot of functions in my game that would be used by multiple different windows forms. So instead of simply copy and pasting the functions across all my windows forms I decided to code a class library and link it to my Visual Studio.NET project dependencies. In my class library I developed 2 modules (bodies of code that do not store data), one for string validation and one for math functions along with various other classes I thought would be useful. I researched by reading an article on how to dynamically link the library to my pilot software and later my final software.[7] I did this by clicking 'Project>Add Reference…' and then browsing for the '.dll' file in my class library's debug folder. In the *MathFunc* module I coded some useful methods I used in my main software such as: a discrete random number function and a recursive quicksort procedure.

When writing and testing my classes I ran into various problems and realised the techniques I was using could be improved. During the development of my pilot software I developed 4 basic classes illustrated in *Figure 3.1*: Deck, Card, Hand and Cardholder. These classes were all vital for keeping my code flexible and readable, so any change in direction of my design could easily be achieved. Initially most of the members of my classes were public, meaning their values could be accessed outside the class. This was because I did not know that you could define a variable as a specific class type without instantiating it. Therefore, all my global variables were calling the constructor of their class as soon as they were defined. This meant I had to keep the constructors of my classes empty and later set that values of my global variables in the *Form.Load* event. I realised this was a poor coding technique after researching the

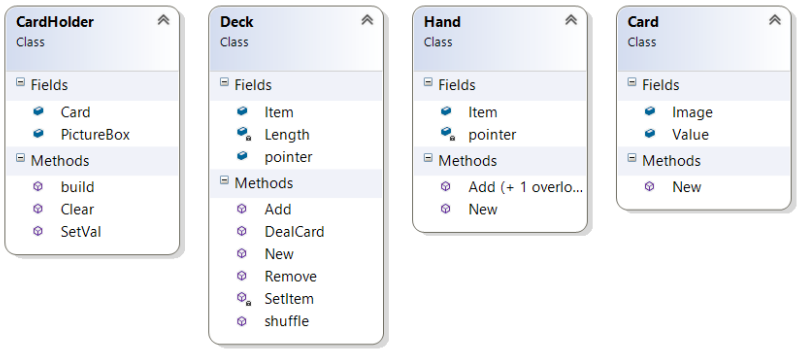


Figure 3.0 - Pilot software class diagram

[5] "Managing Groups Of Objects In Visual Basic". *Msdn.Microsoft.Com*. Last modified 2017. Accessed February 16, 2017. https://msdn.microsoft.com/en-us/library/495te598(v=vs.100).aspx.

[6] "Addhandler Statement". *Docs.Microsoft.Com*. Last modified 2017. Accessed February 19, 2017. https://docs.microsoft.com/en-us/dotnet/visual-basic/language-reference/statements/addhandler-statement.

[7] "Adding DLL Reference To VB.NET Project". *Stackoverflow.Com*. Last modified 2017. Accessed February 30, 2017. https://stackoverflow.com/questions/11674944/adding-dll-reference-to-vb-net-project.

idea of encapsulation.[8] Encapsulation, is the idea of keeping your members private and writing public methods to set and get the members of the class. I ran into multiple fatal errors during my pilot software: 'Object reference not set to an instance of an Object' and 'Null Pointer Exception'. This was the main turning point in developing my skills and I later rectified my coding techniques in my final piece of software, writing 'setter' and 'getter' methods along with writing relevant constructors with appropriate parameters. I was also developing an understanding of inheritance, whereby a class inherits its super class's protected methods and members. I used the idea of inheritance and polymorphism to later plan and design my classes for my final piece of software (See Appendix C for a detailed plan on classes).

Furthermore, I experimented with some visual effects where *PictureBoxes* moved if you hovered over them. This was a mechanic from *Hearthstone* that I liked and considered including in my main project. However, I decided not to include it in my main project because the *PictureBoxes* did not move smoothly and appeared glitchy which was down to the fact that Visual Studio.NET does not support visual effects very well. However, the mechanic I did include from *Hearthstone,* was another hovering mechanic, whereby a larger display of a card pops-up when you hover over it. To implement this in my game I researched into the *Cursor.Position* property to fetch the position of the cursor in windows forms.[9] This allowed me to then display a larger version of the card at the location I had just fetched. I finally moved on from my pilot software after I figured fixing all the fatal errors would be too time consuming and not worth it. This was because the problems were woven within the design of the software. I moved on to my main project and started planning the classes I would have to code, taking into consideration the mechanics I would include, and bearing in mind to be very methodical; to avoid any errors by taking my time and evaluating my code at every opportunity.

During the development of my main project, the research continued. However, the research was mainly about configuring certain controls and how to solve generic problems in windows forms, rather than OOP, because at this point my understanding of OOP had significantly improved. One thing I researched into and used in my project a great deal were Enums or enumeration.[10] Enumeration is a technique which defines a variable that can only take a finite set of values. This makes code more readable and makes developing software easier, because you can see the list of values that a variable can take when you are coding, along with easily adding to this list of values. For example, I used enumeration for the variable *Champion Name*. There are only 4 champions (characters) you can choose to play as in my game. Therefore, enumeration defines the variable *Champion Name* with the 4 valid names that this variable can take. At run-time I have 2 instances of this variable one for the user and one for the AI. enumeration was very helpful for understanding my code, making it flexible and developing switch statements in which an Enum value was used as the selection variable. A consistent problem I ran into multiple times during the development of my software was forms and *PictureBoxes* flashing or glitching when they are shown. I assumed that this was because Visual Studio.NET does not support visual effects and graphics very well. I researched a lot to find a solution for this problem; using *DoubleBuffered = True* was suggested.[11] However, I used timers in my software to fade in and fade out *PictureBoxes* or forms that pop up, allowing the game to run fluidly and removing the flashing problem. I originally used global variables to send data from starting the *Timer* to the *Timer.Tick* event. However, after researching into the *Timer* class I

---

[8] "Introduction To Objects In Visual Basic". *Msdn.Microsoft.Com*. Last modified 2017. Accessed January 18, 2017. https://msdn.microsoft.com/en-us/library/zztsbwsx(v=vs.90).aspx.

[9] "Cursor.Position Property (System.Windows.Forms)". *Msdn.Microsoft.Com*. Last modified 2017. Accessed February 30, 2017. https://msdn.microsoft.com/en-us/library/system.windows.forms.cursor.position(v=vs.110).aspx.

[10] "Enum Statement (Visual Basic)". *Docs.Microsoft.Com*. Last modified 2017. Accessed March 29, 2017. https://docs.microsoft.com/en-us/dotnet/visual-basic/language-reference/statements/enum-statement.

[11] "How To: Reduce Graphics Flicker With Double Buffering For Forms And Controls". *Docs.Microsoft.Com*. Last modified 2017. Accessed March 31, 2017. https://docs.microsoft.com/en-us/dotnet/framework/winforms/advanced/how-to-reduce-graphics-flicker-with-double-buffering-for-forms-and-controls.

discovered the *Timer* control in windows forms has a member called *Tag* which can store an object.[12] I then used *Timer.Tag* to send data, rather than using a global variable, which was much better practice. I wanted to have more experience working with complicated windows forms controls so I decided to use the *DataGridView* control in windows forms for displaying the deck of cards in the start-up form. I learnt how to remove items when handling the *CellMouseClick* event by retrieving the *RowIndex* from the *DataGridViewCellMouseEventArgs*[13]. I also researched how to suppress the *DataGridView* selection by clearing the selection in the *SelectionChanged* event,[14] and adding to the *DataGridView* using *Rows.add(row).*[15] I further researched into some more windows forms specific solutions, loading a word document file into a *RichTextBox* control by saving the word document as an rtf (rich text format) file.[16] This was required to display the rules of my game and all the software references. I also researched how to play sound effects with *My.Computer.Audio.play(,).*[17]

This leads on to one of the biggest problems I had towards the end of the project; playing the sound effects over the background music. A solution I found involved using a windows media player control to play music at the same time as using *My.Computer.Audio.play(,)*[18]*.* I tried playing the background music on a windows media player control, using a file from my resources directory. However, this was not possible because the windows media player control would only play a file from a file path not a memory stream. I pieced together a solution I found for writing data to a file from resources[19], and a solution for creating a file,[20] developing the solution shown in *Figure 4.0*.

```
Private Sub Background_Music_Player_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    Try
        Me.AxWindowsMediaPlayer1.settings.volume = 30
        Me.Owner = My.Forms.Launcher
        ' file path set to the application directory
        Dim path As String = My.Application.Info.DirectoryPath + "\back_groundmusic.wav"

        ' Create or overwrite the file.
        Dim fs As IO.FileStream = IO.File.Create(path)
        Dim myMemStream As New System.IO.MemoryStream
        My.Resources.background_music.CopyTo(myMemStream)
        fs.Close()
        Dim myBytes() As Byte = myMemStream.ToArray

        ' Add data to the file.
        My.Computer.FileSystem.WriteAllBytes(path, myBytes, False)

        ' play the music on the windows media player object
        Me.AxWindowsMediaPlayer1.URL = path
        Me.AxWindowsMediaPlayer1.settings.setMode("loop", True)
        Me.AxWindowsMediaPlayer1.Ctlcontrols.play()
    Catch ex As Exception
    End Try
End Sub
```

Figure 4.0 - Playing memory stream from resources on windows media player

Another problem I had was running the

[12] "Timer.Tag Property (System.Windows.Forms)". *Msdn.Microsoft.Com*. Last modified 2017. Accessed April 4, 2017. https://msdn.microsoft.com/en-us/library/system.windows.forms.timer.tag(v=vs.110).aspx.

[13] "Delete Row From Datagridview Via Button On The Row". *Stackoverflow.Com*. Last modified 2017. Accessed June 25, 2017. https://stackoverflow.com/questions/31528768/delete-row-from-datagridview-via-button-on-the-row.

[14] "Disable Cell Highlighting In A Datagridview". *Stackoverflow.Com*. Last modified 2017. Accessed June 25, 2017. https://stackoverflow.com/questions/1745272/disable-cell-highlighting-in-a-datagridview.

[15] "Datagridview.Rows Property (System.Windows.Forms)". *Msdn.Microsoft.Com.* Last modified 2017. Accessed June 25, 2017. https://msdn.microsoft.com/en-us/library/system.windows.forms.datagridview.rows(v=vs.110).aspx.

[16] "VS 2010 [RESOLVED] Loading Word Document In Richtextbox For Help Menu Contents-Vbforums". *Vbforums.Com*. Last modified 2017. Accessed June 30, 2017. http://www.vbforums.com/showthread.php?688839-RESOLVED-Loading-word-Document-in-RichTextBox-for-Help-menu-contents.

[17] "Playing Sounds (Visual Basic)". *Docs.Microsoft.Com*. Last modified 2017. Accessed June 4, 2017. https://docs.microsoft.com/en-us/dotnet/visual-basic/developing-apps/programming/computer-resources/playing-sounds.

[18] "Simplest Way To Play 2 Or More Sounds At The Same Time.". *Social.Msdn.Microsoft.Com*. Last modified 2017. Accessed June 6, 2017. https://social.msdn.microsoft.com/Forums/vstudio/en-US/3421d6dc-587a-4acf-81d2-0074284c720d/simplest-way-to-play-2-or-more-sounds-at-the-same-time?forum=vbgeneral.

[19] "Write File With Text From Resources Visual Basic". *Stackoverflow.Com*. Last modified 2017. Accessed July 7, 2017. https://stackoverflow.com/questions/28117533/write-file-with-text-from-resources-visual-basic.

[20] "Filesystem.Writeallbytes Method (Microsoft.Visualbasic.Fileio)". *Docs.Microsoft.Com*. Last modified 2017. Accessed July 7, 2017. https://docs.microsoft.com/en-us/dotnet/api/microsoft.visualbasic.fileio.filesystem.writeallbytes?view=netframework-4.7.

published version of my software on other computers. The controls in the windows forms were not scaling appropriately to the resolution of the screen, so they were either too small or too big making the game unplayable. To solve this, I had to write a short procedure to scale each of the controls by changing their sizes and their locations, according to the resolution of the screen. I called this procedure in the constructors of some of my classes so I did not have to call this procedure for absolutely all the controls. Finally, I learnt some more basic VB.NET (programming language) syntax such as: using *Me.myvariable* so that a parameter of a function or procedure can have the same identifier as a member of a class, along with defining a function's return type as a specific class which is not obligatory in VB.NET.

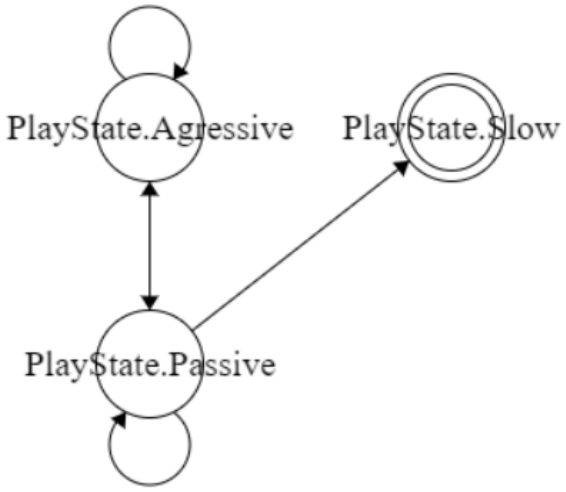


Figure 5.0 - Finite state machine of AI

For the final part of my project I had planned to research and develop an AI to play against the player in my game. I read an article on designing and implementing game AIs which was very useful.[21] It first outlined the difference between an AI for a game and the general preconception of what an AI is and what it does; for games, the AI needs to simulate intelligence by making good decisions that seem logical to the real players, as opposed to being self-aware and have adaptive learning capabilities. Given the AI does not need to consider anything more than playing the game, the complexity of the AI is dictated by the complexity of the game. I started off by building a simple rule based AI, one of the simplest forms of AI. A rule based AI makes decisions based on information given to it by the game. As, in the context of a card game, information is static I used an entity pull system. This means the AI pulls relevant information from the game when it has to play, it then makes a decision and responds immediately. To further develop the AI's strategic approach, I made it occupy 3 different states shown in *Figure 5.0*. The AI makes different decisions based on the same information depending on which state it is in. In my case information about the opponent governs the state of the AI and information about itself governs its ultimate decision making. Implementing these states make it a complex AI, however being a purely rule based system it is not adaptive and thus has room for improvement. If I were to connect the AI to a database, this could be possible.

In terms of skills development my project was a remarkable success; my ability in OOP developed more than I could imagine. I learnt the importance of encapsulating my classes and applying the ideas of inheritance and polymorphism to create efficient and flexible classes. I also now fully understand the potential and power of this coding paradigm and can see how it can be applied to many different situations. I have already seen further improvements coding my A-Level Computer Science project where I am coding in Java; a more demanding OOP language. Along with my development in OOP, I learnt the fundamentals for developing a piece of software in windows forms; various little tricks and solutions, how to configure and work with a variety of windows forms controls along with understanding the importance of keeping all my files in the resources folder. I feel very comfortable using Visual Studio.NET and all this has given me a lot of confidence in developing more software using windows forms. I have also developed a basic understanding of the several ways to design an AI, which I believe will be beneficial to me in the future. This project has

[21] Donald Kehoe. "Designing Artificial Intelligence For Games (Part 1) | Intel® Software". *Software.Intel.Com*. Last modified 2017. Accessed June 6, 2017. https://software.intel.com/en-us/articles/designing-artificial-intelligence-for-games-part-1.

also helped with my time management and planning skills along with giving me a good insight into what it takes to complete a large project.

Looking back, my project ran smoothly from start to finish with most of my own personal deadlines being completed on time. I constantly referred to my Gantt chart throughout the project. However, with some tasks I underestimated the time I would have to dedicate to them. My primary research into popular CCGs, was very successful as I could work out what I should include in my game from things such as basic mechanics and the ruleset (See Appendix A), to visual effects and sounds along with the standard layout I tested and later used in my final project (See Appendix B). My pilot software was also very useful as it allowed me to research how to configure and use various windows forms controls. I then started on a blank windows forms on my final piece of software and used what I had learnt to design the layout of the game and develop some fundamental classes. Using the experience from my pilot software, I could plan 14 of the classes I would have to develop (See Appendix C). However, because of moving on from my pilot software 2 weeks earlier than planned I was not as experienced as I had hoped to be. This meant the 5 classes *BoardHolder*, *HandHolder*, *BuffDisplay, AuraDisplay* and *CardGroup* were all classes I had not planned for and wrote half way through development when I realised I needed them. I wrote 4 classes inherited from *CardHolder* because all of them held cards. However, they needed to have different interactions with the user so when I instantiated them I needed to address different events to different procedures. I wrote *CardGroup* when I realised the classes for *Hand* and *PlayerBoard* were very similar and I was writing the same procedures for both. So instead of keeping them separate I reused the code and gave them both a super class. All 20 of the classes I wrote for the game form are illustrated in *Figure 6.0*. Unfortunately, because of these unforeseen classes, giving the game functionality took 3 weeks longer than expected. I believe if I had not run into so many errors in my pilot software I would have been able to spend more time on it, and I would have been able to plan all the classes I would have to write. This would have been very beneficial and sped up the class

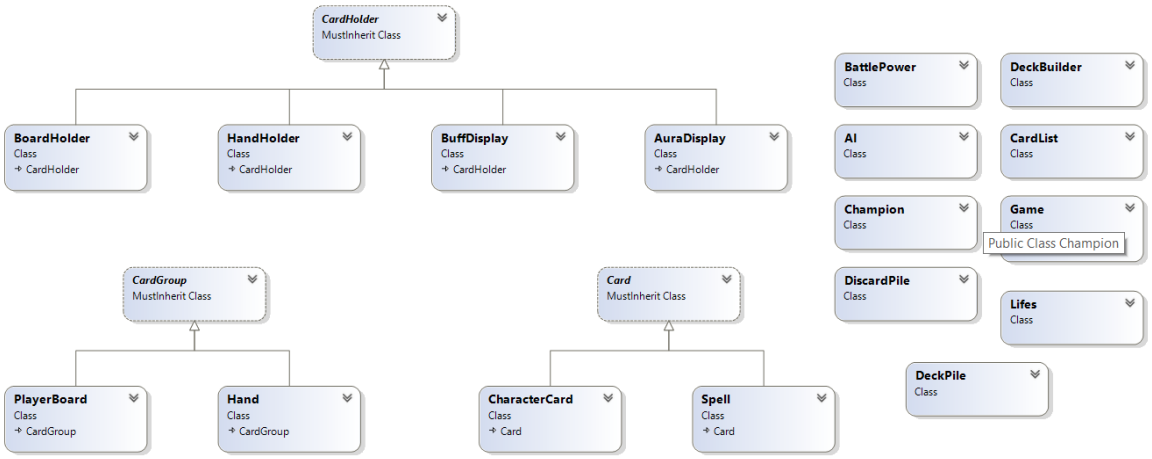


Figure 6.0 - Class diagram of the game

development process by about 2-3 weeks. My next step was to develop a set of cards to play with. Initially I aimed to design 15 cards for each character, using standard ideas from the online CCGs I researched into, and basing a variety of my spell cards around luck. To design my set of cards I sourced all my artwork from 'deviantart.com' a website that allows graphical artists and designers to share their artwork.[22] Any artwork sourced from 'deviantart.com' must be referenced to the user that posted it, so all the artwork I used is fully referenced in my software (See Appendix D for

[22] "Deviantart - The Largest Online Art Gallery And Community". *Deviantart*. Last modified 2017. Accessed April 7, 2017. https://www.deviantart.com/.

software references). I used a card template,[23] images from 4 different artists,[24] and I used the online image editor 'sumopaint.com' to piece together my cards and edit any other artwork I used.[25] *Figure 7.0* is an example of one of the cards in my set. I ended up making a set of 65 playable cards in total. I overestimated the time this would take and I finished this part of my project a week before my deadline, even though the process was still time consuming. Sourcing the images was the primary problem, however designing the cards was quite enjoyable. I then fully developed the game with visual effects, displays, sounds and music. I did various bug fixes (for example some cards were not quite doing the right thing) and tests to make sure the game ran smoothly. I finally redesigned some cards and champion abilities to improve gameplay and make the game more balanced, along with developing the strategic rule based AI. I then developed the start-up screen where you could select which cards to play with and which character to use, along with a simple menu to display the rules of my game and all the software references (Appendix D). When I finished my project, I was 2 weeks behind on my schedule which was not bad considering the scale of my project. The overall result of my project was around 5000 lines of code, 16 different windows forms and 20 classes just for the game form. The project was far bigger than I had anticipated and took an estimated 300 hours to complete.

Figure 7.0 - Example of character card

What I was most proud of, is how professional my project looked visually. In my opinion, the graphics of the game were its strongest feature. A lot of feedback from people I tested my software with also praised my software's professional look. It also turned out to be quite an enjoyable game to play; with a combination of a standard *Gwent* style game and some of my own unique rules and mechanics along with well-designed, fun cards to play with. To conclude, if I were to change anything I would try to develop this sort of game in a different compiler; one that supports graphics and video game development better than Visual Studio.NET. I used Visual Studio.NET as it was the compiler I am most comfortable working in, but because of this any big visual effects or animations were severely lacking from my game compared to a professional game. I would have also liked to try to develop the AI further into an adaptive system by connecting it to a database. This would have added another element of challenge and would have really tested the user and me as the developer. I would also take into consideration more planning for any other future OOP projects. I would strongly encourage other people to pursue the Extended Project Qualification, especially if they are looking to take their understanding of a subject or area above the curriculum. I would also advise them to choose a project they know they will enjoy completing, particularly if it is of similar magnitude to mine. This was a very fun project for me to complete and I believe it was a success with respect to my desired outcomes and intentions. But most importantly it was very beneficial in developing my OOP ability and understanding, giving me the ability to evaluate my own code with respect to efficiency, now that I fully understand the ideas of encapsulation, inheritance and polymorphism.

[23] Zoltan-Graphics. *Card Template*. Image, 2017. Accessed April 3, 2017. https://zoltan-graphics.deviantart.com/art/Card-template-501369704.

[24] StuArtStudios et al., Online (https://stuartstudios.deviantart.com/gallery/. http://peterprime.deviantart.com/gallery/. https://thiago-almeida.deviantart.com/gallery/. https://yigitkoroglu.deviantart.com/gallery/., 2017).

[25] "Sumopaint.Com". *Sumopaint.Com*. Last modified 2017. Accessed April 10, 2017. https://www.sumopaint.com/home/.

Appendix A

Card Game Instructions

Basic Rules

- Each player has a deck of 30 cards.
- At the start of each game you draw 10 random cards from your deck.
- You have the option to redraw up to 3 cards before the game starts.
- Each game consists of up to 3 rounds.
- The first player to win 2 rounds wins the game (best of 3 format).
- The round ends when both players have either passed or run out of cards available.
- The player who wins the round is the player with the highest battle power at the end of the round (indicated above your champion display during the game).
- You may pass any time during your turn, just press the end turn button.
- At the beginning of each new round you draw 1 card.
- You may only hold 10 cards in your hand. Any additional cards overdrawn will be placed in your discard pile.
- Each player can only play one card each turn or may choose to use their champion ability.

Choosing Your Champion

- Each champion has a different champion ability that you may use whenever you like, during the game.
- You cannot use your champion ability last as your turn will be ended.
- Hover over the champion picture to read the ability.
- Select a champion by ticking the associated check box next to its picture.
- To use your champion ability double-click your champion display in game, you may also hover over it to re-read its description

Building Your Deck

- Your deck must contain exactly 30 cards.
- Your deck may not have any more than 2 copies of each card.
- Your deck may only contain 1 copy of each special or golden card.
- All golden character cards are champion specific, meaning only a specific champion can include each golden character card in their deck.
- Good decks use a good balance of spell and character cards.
- You may choose to use a pre-set deck, which is a deck designed by the developer for each champion, to help you start off.
- To build your deck, click the card buttons to add to your deck, to remove click the buttons inside the deck container.
- You may hover over the card buttons to view the card

Types of Cards

- Character Card – each character card has a battle power value. When a character card is played it is played to your battlefield and its battle power value is added to your overall battle power. Each character card also has two other variables. Its species and its type. A character card's species can range from: Human, Dragon, Spirit or Fairy. And its type can range from: Dark, Balance, Law and Nature. Various spells interact with character card's species and types so be aware when designing your deck.
- Golden Card – each champion has various golden character cards associated with it. Only this champion can include these golden cards in its deck. Only one copy of a golden card can be used in a deck. Golden cards are a type of character card. However, once played they cannot be killed or revived from the discard pile in a later round.
- Spell – A card that is not played onto your battlefield and thus does not have a battle power value. A spell does something indicated in the card description and often interacts with character cards that have been played.
- Instant – A type of spell that does something in one instant indicated in the description, such as: add 50 battle power.
- Buff – A type of spell that is played into the buff slot on the right of your battlefield, it increases your battle power based on the cards in your battlefield. Buff can only be replaced by you and are removed at the end of each turn.
- Aura – A type of spell similar to that of a buff, yet usually weaker and played into the aura slot of the battlefield. There may only be one aura at any one time and thus an aura can be replaced by your opponent's aura card at any time. However, an aura does not get removed at the end of every turn and thus will remain active the whole game if not replaced.

Miscellaneous Information

- You can surrender or quit at any time during a game via the options menu in the top left corner.
- You can mute the music in the top left corner during the game or in the launcher.
- Spells can be placed in the discard pile if overdrawn. However, revive will only revive the top non-golden character card.
- Replenish is the only special spell card. Even though it does not appear golden it can only be used once in each deck. However, it can be used by every champion.
- If you are confused on what a card does hover over it for 2 seconds and a larger display of it will appear in game.
- Spells your opponent plays will be displayed briefly in the centre of your screen make sure you're paying attention.
- To play a card from your hand double click it a single click will do nothing.

Basic Tactics

- If you have a higher battle power and your opponent has passed, it is ideal to pass under most circumstances otherwise you are just wasting cards.
- If your opponent is winning by a large margin and you have 2 lives left, it is usually smart to pass.
- Dead Spells (spells that won't do anything under the circumstances) can be smartly played first to disguise to your opponent what you plan on doing by effectively skipping your turn.

- If you have no spells that draw cards at the start of the game, it is a smart to play less aggressively and hope you get some in the later rounds. When discarding your cards at the beginning of the game it is smart to discard any duplicate spells (not including spells that draw cards) and any weak character cards (with less than 100 battle power).

Figure 8.0 - Card overview

Appendix B

Research and Layout notes

Hearthstone

- End turn button to pass the round. Right side of the screen.
- Hovering mechanics. Mouse enter triggers PictureBox to move up slightly and Mouse leave triggers PictureBox to move back down. Mouse enter also triggers a timer to start. When the timer has stopped an additional PictureBox is shown displaying a larger version of the card.
- Static number of places available for the cards. Hand can only hold 10 cards at any one time (place overdrawn cards in the discard pile). Board can also only hold 10 cards for each player at any one time. If a Card is clicked on and there are no spaces on the board available the card will not be played.
- Consider a simple menu in the corner with similar functions to Hearthstone's menu. Ability to forfeit the game, close the game and consider some other functions.
- A card is selected to be redrawn rather than redrawn immediately when clicked.

Gwent

- Linear layout of the cards, cards displayed on a rectangle board. Board consists of 1 section with 10 place holders rather than 3 sections with an infinite amount of place holders.
- Hand also held in a linear format like the board.
- Champion ability played by clicking on the image of your champion. However, this is displayed on the right of the screen.
- PictureBoxes displaying deck and discard pile also on the right of the screen.
- Buff and Aura cards displayed on the right of the screen. Similar to weather cards in Gwent, however there is a PictureBox for each Buff and Aura instead of one container.
- Visual effects notifying the player when someone passes their turn.
- 2 PictureBoxes to show the lives of each player.

Other Layout elements

- PictureBox for displaying whose turn it is to play.
- PictureBox for each player displaying if they have passed or not.
- Versus loading screen showing the champion you are playing against and their ability so you can re-draw cards accordingly.
- Labels above discard card and Deck showing the number of cards within them. And separate labels showing what they are displaying (Deck or Discard Pile)

Appendix C

Class Plan Overview

AI – class that constitutes the AI playing against the user playing the game. Further research is required to plan this class. Perhaps write simple methods selecting a random card to play and responding when required. This will be implemented later when game is functional.

BattlePower – class handles the battle power or score of the 2 players throughout the game. Class stores an integer and a label reference. When power is updated, the number displayed on the label is updated.

Card – super class of CharacterCard and Spell, stores image, ID and special (boolean). Write methods to return these values.

CardList – class that programmatically instantiates every card being used in the game. Consider using a database rather than hard coding the values for each card. However, this might be necessary for the software to run on other systems.

CardHolder – stores an instance of a Card and a PictureBox reference. Write methods to get the card instance, remove the card (make equal to nothing), and add a card reference that also updates the image of the PictureBox.

Champion – class that handles the champion of a player. Stores the champion (perhaps use enum) picturebox displaying the champion image and ability used (boolean). Methods to get ability and champion.

CharacterCard – inherits Card, stores 3 identifiers: type, species and attack_value. Consider using Enums for identifiers. Write methods to return these values.

DeckBuilder – class that has 1 method. Builds a deck of cards based on what champion you have picked.

Deck – class that acts like a stack. Stores cards in a collection along with a PictureBox reference that displays the deck. Write methods to shuffle the collection, draw a card and shuffle a card back into the collection.

DiscardPile – class that handles the discard pile. Stores a ‘stack’ that handles all the cards in the discard pile. Stores a PictureBox reference that displays the top card of the stack. Write methods to add and remove from discard pile.

Hand – stores a collection of 10 cardholders. Write methods to get cards from cardholders, remove cards, add cards and get PictureBox references. Write algorithm to shift cards when a card has been removed, meaning no gaps in the collection.

Lifes – class that handles the lives of a player. Stores the lives as an integer and 2 PictureBox references one for each life. Write method to remove a life and get the number of lives.

Board – stores collection of 10 card holders. Write methods to get cards from cardholders, remove cards, add cards and get PictureBox references. Write algorithm to shift cards when a card has been removed, meaning no gaps in the collection. Consider writing a super class for both Board and Hand.

Spell – inherits Card. Stores 2 identifiers: type of spell and its effect. Highly recommend using Enums to list the different cards effects.

Appendix D

Software References

Card Art References

Aether Dragon Card Art - PeterPrime. *Forest Dragon*. Image, 2012. Accessed April 10, 2017. https://peterprime.deviantart.com/art/Forest-Dragon-312400777.

Akrambuela Card Art - yigitkoroglu. *Akrabuamelu*. Image, 2017. Accessed April 29, 2017. https://yigitkoroglu.deviantart.com/art/Akrabuamelu-668912907.

Alleyn Card Art - PeterPrime. *Ares*. Image, 2013. Accessed April 10, 2017. https://peterprime.deviantart.com/art/Ares-349172666.

Aris Card Art - yigitkoroglu. *Legend Of The Cryptids - Vampire Hunter*. Image, 2013. Accessed April 28, 2017. https://yigitkoroglu.deviantart.com/art/Legend-of-the-Cryptids-Vampire-Hunter-351797080.

Autumn Dragon Card Art - PeterPrime. *Autumn Dragon*. Image, 2014. Accessed April 10, 2017. https://peterprime.deviantart.com/art/Autumn-Dragon-509559907.

Blood Knight Card Art - yigitkoroglu. *Legend Of The Cryptids - Vampire Hunter Advanced*. Image, 2013. Accessed April 13, 2017. https://yigitkoroglu.deviantart.com/art/Legend-of-the-Cryptids-Vampire-Hunter-Advanced-351805646.

Character Card Border - Zoltan-Graphics. *Card Template*. Image, 2017. Accessed April 3, 2017. https://zoltan-graphics.deviantart.com/art/Card-template-501369704.

Curse Card Art - yigitkoroglu. *Contact*. Image, 2013. Accessed April 29, 2017. https://yigitkoroglu.deviantart.com/art/Contact-346513778.

Dark Dragon Card Art - PeterPrime. *Black Dragon Tempest*. Image, 2014. Accessed April 5, 2017. https://peterprime.deviantart.com/art/Black-Dragon-Tempest-466401924.

Dark Warden Card Art - StuArtStudios. *Dragonmancer*. Image, 2014. Accessed May 12, 2017. https://stuartstudios.deviantart.com/art/DragonMancer-452850435.

Darklurker Card Art - PeterPrime. *Shadow Lurker*. Image, 2014. Accessed April 18, 2017. https://peterprime.deviantart.com/art/Shadow-Lurker-473318606.

Dead King's Ivy - thiago-almeida. *Mr.Ivy*. Image, 2012. Accessed May 10, 2017. https://thiago-almeida.deviantart.com/art/Mr-Ivy-324568705.

Deeproot Card Art - yigitkoroglu. *Deeproot 4*. Image, 2014. Accessed April 31, 2017. https://yigitkoroglu.deviantart.com/art/Deeproot-4-481083456.

Desmos Card Art - StuArtStudios. *Deamos The Devourer*. Image, 2014. Accessed May 17, 2017. https://stuartstudios.deviantart.com/art/Deamos-the-Devourer-457446968.

Dr. DeadLift Card Art - StuArtStudios. *Deadlift Druid Stuharrington*. Image, 2015. Accessed May 10, 2017. https://stuartstudios.deviantart.com/art/Deadlift-Druid-StuHarrington-516628623.

Dragon Commander Card Art - PeterPrime. *Dragon Lord*. Image, 2014. Accessed April 2, 2017. https://peterprime.deviantart.com/art/Dragon-Lord-428077526.

Dragon Dominion Card Art - yigitkoroglu. *Undead Dragon Rider*. Image, 2013. Accessed April 31, 2017. https://yigitkoroglu.deviantart.com/art/Undead-dragon-rider-365816448.

Dragon Paladin Card Art - PeterPrime. *Dragon Rider Captain*. Image, 2012. Accessed April 2, 2017. https://peterprime.deviantart.com/art/Dragon-Rider-Captain-340169085.

Dragon Warlord Card Art - PeterPrime. *Dragon Overlord*. Image, 2017. Accessed April 13, 2017. https://peterprime.deviantart.com/art/Dragon-Overlord-323184630.

Drakthar Card Art - yigitkoroglu. *Edimmu*. Image, 2017. Accessed April 24, 2017. https://yigitkoroglu.deviantart.com/art/Edimmu-666848615.

Embody Card Art - yigitkoroglu. *DW 5 God Of Nightmares*. Image, 2011. Accessed April 19, 2017. https://www.deviantart.com/art/DW-5-God-of-Nightmares-210235460.

Endarken Card Art - yigitkoroglu. *The Cursed Throne*. Image, 2016. Accessed April 25, 2017. https://yigitkoroglu.deviantart.com/art/The-Cursed-Throne-649377884.

Enforcement Card Art - yigitkoroglu. *Ritual Of The Old Ones*. Image, 2016. Accessed April 11, 2017. https://yigitkoroglu.deviantart.com/art/Ritual-of-the-Old-Ones-651522782.

Ennrith Card Art - PeterPrime. *Forest Lord*. Image, 2012. Accessed April 28, 2017. https://peterprime.deviantart.com/art/Forest-Lord-331946563.

Ennrith Unleashed Card Art - PeterPrime. *Forest Lord*. Image, 2012. Accessed April 28, 2017. https://peterprime.deviantart.com/art/Forest-Lord-331946563.

Ent Archer Card Art - StuArtStudios. *Ent Archer*. Image, 2013. Accessed May 10, 2017. https://stuartstudios.deviantart.com/art/Ent-Archer-353732399.

Ent Shaman Card Art - thiago-almeida. *Treefolk Shaman*. Image, 2012. Accessed May 10, 2017. https://thiago-almeida.deviantart.com/art/treefolk-Shaman-286342563.

Equilibrium Card Art - yigitkoroglu. *Azazel*. Image, 2010. Accessed April 31, 2017. https://yigitkoroglu.deviantart.com/art/Azazel-167264648.

Etheral Sacrifice Card Art - StuArtStudios. *A Titan's Duel*. Image, 2015. Accessed May 12, 2017. https://stuartstudios.deviantart.com/art/A-Titan-s-Duel-541292050.

Executioner Card Art - yigitkoroglu. *Executioner*. Image, 2010. Accessed April 29, 2017. https://yigitkoroglu.deviantart.com/art/Executioner-162356043.

Fairy Dragon Card Art - PeterPrime. *Fairy Dragon*. Image, 2013. Accessed April 10, 2017. https://peterprime.deviantart.com/art/Fairy-Dragon-379870443.

Fox Archer Card Art - StuArtStudios. *Robin Hood*. Image, 2017. Accessed May 10, 2017. https://stuartstudios.deviantart.com/art/Robin-Hood-674270719.

Gautzelin Card Art - PeterPrime. *Beatrix And Wagner Santos*. Image, 2014. Accessed April 18, 2017. https://peterprime.deviantart.com/art/Beatrix-and-Wagner-Santos-424944275.

Great Ent Card Art - thiago-almeida. *Druid*. Image, 2013. Accessed May 10, 2017. https://thiago-almeida.deviantart.com/art/Druid-389043462.

Hallowed Warlord card Art - yigitkoroglu. *Lazy Knight*. Image, 2011. Accessed April 31, 2017. https://yigitkoroglu.deviantart.com/art/lazy-knight-200863580.

Hell Drake Card Art - PeterPrime. *Hell Dragon*. Image, 2014. Accessed April 10, 2017. https://peterprime.deviantart.com/art/Hell-Dragon-477471345.

Jadirrayis Card Art - PeterPrime. *Dragon Apple?*. Image, 2013. Accessed April 10, 2017. https://peterprime.deviantart.com/art/Dragon-apple-349058533.

King's Blessing Card Art - yigitkoroglu. *Tales From The Kings Court*. Image, 2016. Accessed April 25, 2017. https://yigitkoroglu.deviantart.com/art/Tales-From-The-Kings-Court-592773417.

Last Stand Card Art - yigitkoroglu. *False Victory*. Image, 2016. Accessed April 25, 2017. https://yigitkoroglu.deviantart.com/art/False-Victory-653442058.

Last Sun Soldier Card Art - yigitkoroglu. *Marduk*. Image, 2017. Accessed April 20, 2017. https://yigitkoroglu.deviantart.com/art/Marduk-668912840.

Lizard Rider Card Art - yigitkoroglu. *Lizard Rider*. Image, 2011. Accessed April 28, 2017. https://yigitkoroglu.deviantart.com/art/Lizard-Rider-200311267.

Mouse Warrior Card Art - StuArtStudios. *Guardian Of Summerhall*. Image, 2014. Accessed May 10, 2017. https://stuartstudios.deviantart.com/art/Guardian-of-SummerHall-455310729.

Nergal Card Art - yigitkoroglu. *Nergal*. Image, 2017. Accessed April 27, 2017. https://yigitkoroglu.deviantart.com/art/Nergal-667612695.

Nergal's Secret Card Art - yigitkoroglu. *Creation Of Hell Part 4*. Image, 2015. Accessed April 29, 2017. https://yigitkoroglu.deviantart.com/art/Creation-of-Hell-Part-4-557673278.

Night Rider Card Art - yigitkoroglu. *Valkrie*. Image, 2011. Accessed April 29, 2017. https://yigitkoroglu.deviantart.com/art/Valkyrie-196398777.

Nullify Aura Card Art - yigitkoroglu. *Life*. Image, 2011. Accessed April 31, 2017. https://yigitkoroglu.deviantart.com/art/Life-273752008.

Nullify Card Art - yigitkoroglu. *Emissary Corwan*. Image, 2014. Accessed April 31, 2017. https://yigitkoroglu.deviantart.com/art/Emissary-Corwan-449343530. Red Tree Blossom Card Art - yigitkoroglu. *Tree*. Image, 2014. Accessed April 31, 2017. https://yigitkoroglu.deviantart.com/art/Tree-449978101.

Overgrowth Card Art - yigitkoroglu. *Widow The Sea Album Cover*. Image, 2014. Accessed April 28, 2017. https://yigitkoroglu.deviantart.com/art/Widow-the-Sea-Album-Cover-441580694.

Pilgrimage Card Art - yigitkoroglu. *Pilgrim*. Image, 2016. Accessed April 31, 2017. https://yigitkoroglu.deviantart.com/art/Pilgrim-612636019.

Recycle Card Art - yigitkoroglu. *Executioners Cowl*. Image, 2014. Accessed April 28, 2017. https://yigitkoroglu.deviantart.com/art/Executioners-Cowl-481083583.

Reinforce Card Art - yigitkoroglu. *Tales From The Kings Court Rough*. Image, 2015. Accessed April 31, 2017. https://yigitkoroglu.deviantart.com/art/Tales-From-The-Kings-Court-Rough-511877462.

Rejuvenate Card Art - yigitkoroglu. *Executioner's Cowl*. Image, 2014. Accessed April 29, 2017. https://yigitkoroglu.deviantart.com/art/Executioner-s-Cowl-451732710.

Replenish Card Art - yigitkoroglu. *Uninvited Guests*. Image, 2016. Accessed April 15, 2017. https://yigitkoroglu.deviantart.com/art/Uninvited-Guests-607364037.

Revive Card Art - yigitkoroglu. *Relic*. Image, 2013. Accessed April 29, 2017. https://yigitkoroglu.deviantart.com/art/Relic-418857557.

Shadow Drake Card Art - PeterPrime. *Forest Dragon*. Image, 2012. Accessed April 13, 2017. https://peterprime.deviantart.com/art/Forest-Dragon-312400777.

Skeleton Mage Card Art - thiago-almeida. *Undead Mage*. Image, 2012. Accessed May 12, 2017. https://thiago-almeida.deviantart.com/art/Undead-Mage-286342400.

Spell Card Border/Overlay - Zoltan-Graphics. *Card Template*. Image, 2017. Accessed April 3, 2017. https://zoltan-graphics.deviantart.com/art/Card-template-501369704.

Spirit Warrior Card Art - yigitkoroglu. *Milkman*. Image, 2016. Accessed April 29, 2017. https://yigitkoroglu.deviantart.com/art/milkman-635524212.

Tainted Growth Card Art - yigitkoroglu. *Pre DW 5 Entry*. Image, 2010. Accessed April 31, 2017. https://yigitkoroglu.deviantart.com/art/Pre-DW-5-entry-189867620.

The Faceless Druid Card Art - thiago-almeida. *Walking Druid*. Image, 2012. Accessed May 12, 2017. https://thiago-almeida.deviantart.com/art/Walking-Druid-344220280.

Thorkel Asvard Card Art - yigitkoroglu. *Odin Alternative*. Image, 2009. Accessed April 31, 2017. https://yigitkoroglu.deviantart.com/art/Odin-Alternative-121469100.

Tormented Dregs Card Art - yigitkoroglu. *ERESHKIGAL*. Image, 2017. Accessed April 29, 2017. https://yigitkoroglu.deviantart.com/art/ERESHKIGAL-668912876.

Transform Card Art - yigitkoroglu. *Ruin : Awakening Cover*. Image, 2013. Accessed April 31, 2017. https://yigitkoroglu.deviantart.com/art/Ruin-Awakening-Cover-372013587.

Twin Knights Card Art - PeterPrime. *Dark Knight Concept*. Image, 2012. Accessed April 13, 2017. https://peterprime.deviantart.com/art/Dark-Knight-Concept-320994069.

Unleash Nergal Card Art - yigitkoroglu. *Creation Of Hell Part 4*. Image, 2015. Accessed April 29, 2017. https://yigitkoroglu.deviantart.com/art/Creation-of-Hell-Part-4-557673278.

Unleash Wyvern Card Art - yigitkoroglu. *Taming A Dragon*. Image, 2010. Accessed April 28, 2017. https://yigitkoroglu.deviantart.com/art/Taming-a-Dragon-164957871.

Warden Card Art - thiago-almeida. *Druid Wanderer*. Image, 2012. Accessed May 12, 2017. https://thiago-almeida.deviantart.com/art/Druid-wanderer-324567617.

Zyrgym Card Art - PeterPrime. *Great Imperial Dragon Fraener*. Image, 2012. Accessed March 29, 2017. https://peterprime.deviantart.com/art/Great-Imperial-Dragon-Fraener-331237781.

Champion Art References

The Knight Champion Art - PeterPrime. *Praetorian Tercius*. Image, 2012. Accessed April 4, 2017. https://peterprime.deviantart.com/art/Praetorian-Tercius-333483249.

The Lord Champion Art - PeterPrime. *Dragon Lord*. Image, 2013. Accessed April 1, 2017. https://www.deviantart.com/art/Dragon-Lord-356094066.

The Sage Champion Art - PeterPrime. *Druid Concept*. Image, 2013. Accessed April 10, 2017. https://peterprime.deviantart.com/art/Druid-concept-370026553.

The Witch Champion Art - PeterPrime. *Dark Knight*. Image, 2013. Accessed April 14, 2017. https://peterprime.deviantart.com/art/Dark-Knight-409871021.

UI Element References

Credits/References Button UI - Waterman, Johnny. *Kingsroad UI Elements*. Image, 2014. Accessed May 16, 2017. https://dribbble.com/shots/1588952-KingsRoad-UI-Elements.

End Turn UI Button – Goodall, Alex. *End Turn Button*. Image, 2017.

Exit Button UI - Waterman, Johnny. *Kingsroad UI Elements*. Image, 2014. Accessed May 16, 2017. https://dribbble.com/shots/1588952-KingsRoad-UI-Elements.

In Game Menu Background - Waterman, Johnny. *Kingsroad UI Elements*. Image, 2014. Accessed May 16, 2017. https://dribbble.com/shots/1588952-KingsRoad-UI-Elements.

Instructions Button UI - Waterman, Johnny. *Kingsroad UI Elements*. Image, 2014. Accessed May 16, 2017. https://dribbble.com/shots/1588952-KingsRoad-UI-Elements.

Music Button UI - Waterman, Johnny. *Kingsroad UI Elements*. Image, 2014. Accessed May 16, 2017. https://dribbble.com/shots/1588952-KingsRoad-UI-Elements.

Popup Message Background - Waterman, Johnny. *Kingsroad UI Elements*. Image, 2014. Accessed May 16, 2017. https://dribbble.com/shots/1588952-KingsRoad-UI-Elements.

UI Text Button - StrumpyStrust. *UI Button*. Image, 2014. Accessed May 27, 2017. https://opengameart.org/content/ui-button.

Misc. References

Play card Icon - Quoting. *Card Play Icon*. Image, n.d. Accessed May 13, 2017. http://game-icons.net/quoting/originals/card-play.html.

Skull Icon - Lorc. *Sharped Teeth Skull Icon*. Image, n.d. Accessed May 10, 2017. http://game-icons.net/lorc/originals/sharped-teeth-skull.html,

Game Woodland Background - *Fantasy Forest Wallpaper*. Image, n.d. Accessed April 15, 2017. https://wall.alphacoders.com/big.php?i=220544.

Card Holder Background - *League-Of-Legends-Supremacy-Card-Back-Design*. Image, n.d. Accessed April 15, 2017. https://www.pinterest.com/pin/477311260480741793/.

Card Back - Mind-Force. *Hell Style UI*. Image, 2011. Accessed May 16, 2017. https://mind-force.deviantart.com/art/Hell-Style-UI-257368053.

Battle Power Display Background - Mind-Force. *Hell Style UI*. Image, 2011. Accessed May 16, 2017. https://mind-force.deviantart.com/art/Hell-Style-UI-257368053.

Hand Display Background - PRDart. *Valley Of Embers*. Image, 2012. Accessed March 26, 2017. https://prdart.deviantart.com/art/Valley-of-Embers-312931164.

Battle Field Display Background - Brady26. *Battlefield-War-Zone-Fantasy-Art-Wallpaper*. Image, 2014. Accessed March 20, 2017. http://tolas.wikia.com/wiki/File:Battlefield-War-Zone-Fantasy-Art-Wallpaper.jpg.

Cross Overlay in Discard Card Window - Goodall, Alex. *Cross*. Image, 2017.

Loading Screen Image - Swamps Ack (2011). *Loading*. https://swampsack.bandcamp.com/track/loading-game-please-wait.

Versus Display - Pikells. *Vs Logo.Png*. Image, 2016. Accessed June 10, 2017. http://deathbattlefanon.wikia.com/wiki/File:Vs_logo.png.

Dreadmare Logo - PeterPrime. *Dragon Lord*. Image, 2013. Accessed April 1, 2017. https://www.deviantart.com/art/Dragon-Lord-356094066.

Dreadmare Title - Gutteridge, Chris. *Dreadmare*. Image, n.d. Accessed June 14, 2017. http://flamingtext.com/logo/Design-Dracula.

Dreadmare Launcher Background - Steam. *Chainsaw Warrior: Lords Of The Night - Death Of The Aztecs*. Image, n.d. Accessed June 10, 2017. https://steamuserimages-a.akamaihd.net/ugc/870980564497309 55/59F44CADD087F4C675457D9AE18D90D66D1A08AB/.

Discard Card Window Background - Steam. *Chainsaw Warrior: Lords Of The Night - Death Of The Aztecs*. Image, n.d. Accessed June 10, 2017. https://steamuserimages-a.akamaihd.net/ugc/870980564497309 55/59F44CADD087F4C675457D9AE18D90D66D1A08AB/.

Laurel Icon - Lorc. *Laurel Crown Icon*. Image, n.d. Accessed May 10, 2017. http://game-icons.net/lorc/originals/laurel-crown.html,

Sword Icon - Lorc. *Pointy Sword Icon*. Image, n.d. Accessed May 10, 2017. http://game-icons.net/lorc/originals/pointy-sword.html.

Flag Icon - Lorc. *Flying Flag Icon*. Image, n.d. Accessed May 10, 2017. http://game-icons.net/lorc/originals/flying-flag.html.

Grave Icon - Skoll. *Hast Grave Icon*. Image, n.d. Accessed May 13, 2017. http://game-icons.net/skoll/originals/hasty-grave.html.

Sounds

All Sound effects - "RPG Sound Pack". *Opengameart.Org*. Last modified 2011. Accessed June 6, 2017. https://opengameart.org/content/rpg-sound-pack.

In Game Music - Krogh, Aaron. "310 - World Map (Loop)". *Soundcloud*. Last modified 2012. Accessed July 6, 2017. https://soundcloud.com/aaron-anderson-11/310-world-map-loop?in=aaron-anderson-11/sets/rpg-maker-music-loops.

Inspiration

Inspiration for Basic Game Mechanics and Visuals - *GWENT®: The Witcher Card Game*. Windows. https://www.playgwent.com/en: CD PROJEKT S.A, 2017.

Other Inspiration - *Hearthstone*. Windows. https://us.battle.net/hearthstone/en/: Blizzard Entertainment, 2017.

Bibliography

Figure 1.0 - Goodall, Alex. *Gantt chart*. Image, 2017.

1. "Chronicle: Runescape Legends Wiki". *Chronicle: Runescape Legends Wiki*. Last modified 2017. Accessed January 7, 2017. https://chronicle.gamepedia.com/Chronicle:_RuneScape_Legends_Wiki.

   Figure 2.0 - Jagex. Image, 2017. Accessed October 8, 2017. https://mmos.com/editorials/interview-with-james-sweatman-of-jagex-developers-of-chronicle-runescape-legends.

2. Matulef, Jeffrey. "Hearthstone - The Collectible Card Game That Could Convert You". *Eurogamer.Net*. Last modified 2017. Accessed January 8, 2017. http://www.eurogamer.net/articles/2013-03-26-hearthstone-the-collectible-card-game-that-could-convert-you.

3. "Gwent: The Witcher Card Game". *En.Wikipedia.Org*. Last modified 2017. Accessed January 13, 2017.https://en.wikipedia.org/wiki/Gwent:_The_Witcher_Card_Game.

4. "Stack Overflow - Where Developers Learn, Share, & Build Careers". *Stackoverflow.Com*. Last modified 2017. Accessed January 18, 2017. https://stackoverflow.com/.

5. "Managing Groups Of Objects In Visual Basic". *Msdn.Microsoft.Com*. Last modified 2017. Accessed February 16, 2017. https://msdn.microsoft.com/en-us/library/495te598(v=vs.100).aspx.

6. "Addhandler Statement". *Docs.Microsoft.Com*. Last modified 2017. Accessed February 19, 2017. https://docs.microsoft.com/en-us/dotnet/visual-basic/language-reference/statements/addhandler-statement.

7. "Adding DLL Reference To VB.NET Project". *Stackoverflow.Com*. Last modified 2017. Accessed February 30, 2017. https://stackoverflow.com/questions/11674944/adding-dll-reference-to-vb-net-project.

   Figure 3.0 - Goodall, Alex. *Pilot software class diagram*. Image, 2017.

8. "Introduction To Objects In Visual Basic". *Msdn.Microsoft.Com*. Last modified 2017. Accessed January 18, 2017. https://msdn.microsoft.com/en-us/library/zztsbwsx(v=vs.90).aspx.

9. "Cursor.Position Property (System.Windows.Forms)". *Msdn.Microsoft.Com*. Last modified 2017. Accessed February 30, 2017. https://msdn.microsoft.com/en-us/library/system.windows.forms.cursor.position(v=vs.110).aspx.

10. "Enum Statement (Visual Basic)". *Docs.Microsoft.Com*. Last modified 2017. Accessed March 29, 2017. https://docs.microsoft.com/en-us/dotnet/visual-basic/language-reference/statements/enum-statement.

11. "How To: Reduce Graphics Flicker With Double Buffering For Forms And Controls". *Docs.Microsoft.Com*. Last modified 2017. Accessed March 31, 2017. https://docs.microsoft.com/en-us/dotnet/framework/winforms/advanced/how-to-reduce-graphics-flicker-with-double-buffering-for-forms-and-controls.

12. "Timer.Tag Property (System.Windows.Forms)". *Msdn.Microsoft.Com*. Last modified 2017. Accessed April 4, 2017. https://msdn.microsoft.com/en-us/library/system.windows.forms.timer.tag(v=vs.110).aspx.

13. "Delete Row From Datagridview Via Button On The Row". *Stackoverflow.Com*. Last modified 2017. Accessed June 25, 2017. https://stackoverflow.com/questions/31528768/delete-row-from-datagridview-via-button-on-the-row.

14. "Disable Cell Highlighting In A Datagridview". *Stackoverflow.Com*. Last modified 2017. Accessed June 25, 2017. https://stackoverflow.com/questions/1745272/disable-cell-highlighting-in-a-datagridview.

15. "Datagridview.Rows Property (System.Windows.Forms)". *Msdn.Microsoft.Com*. Last modified 2017. Accessed June 25, 2017. https://msdn.microsoft.com/en-us/library/system.windows.forms.datagridview.rows(v=vs.110).aspx.

16. "VS 2010 [RESOLVED] Loading Word Document In Richtextbox For Help Menu Contents- Vbforums". *Vbforums.Com*. Last modified 2017. Accessed June 30, 2017. http://www.vbforums.com/showthread.php?688839-RESOLVED-Loading-word-Document-in-RichTextBox-for-Help-menu-contents.

17. "Playing Sounds (Visual Basic)". *Docs.Microsoft.Com*. Last modified 2017. Accessed June 4, 2017. https://docs.microsoft.com/en-us/dotnet/visual-basic/developing-apps/programming/computer-resources/playing-sounds.

18. "Simplest Way To Play 2 Or More Sounds At The Same Time.". *Social.Msdn.Microsoft.Com*. Last modified 2017. Accessed June 6, 2017. https://social.msdn.microsoft.com/Forums/vstudio/en-US/3421d6dc-587a-4acf-81d2-0074284c720d/simplest-way-to-play-2-or-more-sounds-at-the-same-time?forum=vbgeneral.

19. "Write File With Text From Resources Visual Basic". *Stackoverflow.Com*. Last modified 2017. Accessed July 7, 2017. https://stackoverflow.com/questions/28117533/write-file-with-text-from-resources-visual-basic.

20. "Filesystem.Writeallbytes Method (Microsoft.Visualbasic.Fileio)". *Docs.Microsoft.Com*. Last modified 2017. Accessed July 7, 2017. https://docs.microsoft.com/en-us/dotnet/api/microsoft.visualbasic.fileio.filesystem.writeallbytes?view=netframework-4.7.

Figure 4.0 - Goodall, Alex. *Class Playing memory stream from resources on windows media player*. Image, 2017.

21. Kehoe, Donald. "Designing Artificial Intelligence For Games (Part 1) | Intel® Software". *Software.Intel.Com*. Last modified 2017. Accessed June 6, 2017. https://software.intel.com/en-us/articles/designing-artificial-intelligence-for-games-part-1.

Figure 5.0 - Goodall, Alex. *Finite state machine for AI.* Image, 2017.

Figure 6.0 - Goodall, Alex. *Class diagram of the game*. Image, 2017.

22. "Deviantart - The Largest Online Art Gallery And Community". *Deviantart*. Last modified 2017. Accessed April 7, 2017. https://www.deviantart.com/.

23. Zoltan-Graphics. *Card Template*. Image, 2017. Accessed April 3, 2017. https://zoltan-graphics.deviantart.com/art/Card-template-501369704.

24. StuArtStudios, PeterPrime, thiago-almeda, and yigitkoroglu. Online. https://stuartstudios.deviantart.com/gallery/. http://peterprime.deviantart.com/gallery/. https://thiago-almeida.deviantart.com/gallery/. https://yigitkoroglu.deviantart.com/gallery/., 2017.

25. "Sumopaint.Com". *Sumopaint.Com*. Last modified 2017. Accessed April 10, 2017. https://www.sumopaint.com/home/.

Figure 7.0 - yigitkoroglu, and Zoltan-Graphics. *Legend Of The Cryptids – Vampire Hunter Advanced. Card Template*. Image, 2017. Accessed April 13, 2017. https://yigitkoroglu.deviantart.com/art/Legend-of-the-Cryptids-Vampire-Hunter-Advanced-351805646. https://zoltan-graphics.deviantart.com/art/Card-template-501369704.

Figure 8.0 – PeterPrime, and Zoltan-Graphics *Dragon Lord*. Card Template. Image, 2014. Accessed April 2, 2017. https://peterprime.deviantart.com/art/Dragon-Lord-428077526. https://zoltan-graphics.deviantart.com/art/Card-template-501369704.