How can a Bespoke Authoring Tool Aid and Assist Board Game Creators in the Development of Augmented Reality Board Games?

Andrew McFadyen
4206010
School of Computer Science
University of Nottingham
psyam6@nottingham.ac.uk

Amoul Mangat
4218547
School of Computer Science
University of Nottingham
psyam7@nottingham.ac.uk

Ian Richards
4203444
School of Computer Science
University of Nottingham
Psyir@nottingham.ac.uk

ABSTRACT

Board game creators are currently not taking advantage of the immense possibilities that Augmented Reality (AR) could add to their products. The current selection of AR authoring tools does not suit the needs of your average board game creator. Yet recent popularity around the few AR Board games that are in existence suggest that there is gap in the board game market. We compare existing AR authoring tools and find that one of the reasons for this is that in order to develop for AR a substantial level of programming expertise is required. We propose and develop a novel set of tools, aimed specifically at the creation AR board games for those without a software development background. The aim is to determine whether or not such a toolkit would be beneficial to current board game creators wanting to implement AR elements into their games. We critically compare our toolkit against a more established authoring application and invite participants to use our toolkit and create AR board games. We record observations about how each of them interacts with our software and use this data to form our conclusion. An authoring tool developed specifically for non-programmers to create AR board games can be hugely beneficial in simplifying and speeding up the development process. However, it is not feasible to create an abstract toolkit capable of providing enough configuration to describe all board games, particularly the more complex. A toolkit of this nature cannot completely replace more code heavy solutions.

CCS CONCEPTS

• Human-centered computing • Human centered computing~mixed / augmented reality

KEYWORDS

Augmented Reality, Authoring tool, Toolkit, Board Games

1. INTRODUCTION

Augmented Reality(AR) is a technology that superimposes computer generated virtual information onto a user's physical environment in real time. The result being a modified reality for the user in which digital content enhances and adds interactivity to the real world [3]. In contrast to Virtual Reality, in which a totally artificial environment is created, AR is designed for real and virtual information to coexist in the same field of view. An important measure of an AR system is how realistic the augmentations are implemented into the physical world. Commonly, graphical objects are projected on top of a camera view, object tracking and data from the accelerometer can then be used to make the object appear to exist in the physical space.

A study from IDC in 2014 looked at finding out how many professional software developers or hobbyist developers exist in the world [2]. There was found to be approximately twenty nine million ICT-Skilled workers in total, of which only around eleven million were software developers and a further seven million were considered hobbyist developers. An ICT-Skilled worker is defined as someone whose occupation involves ICT but is not necessarily a software developer. When compared to the 155 million video game players in the US alone [9] it shows that there is a huge disparity between the number of people playing video games and those who are capable of creating them.

Although practical AR systems has been around since the nineties only recently has consumer technology progressed to a point at which AR applications and hardware is being developed for the mainstream. An everyday smartphone now has the computational power to process comparably realistic AR and recent product launches such as Google Glass and the Microsoft Hololens suggest that AR is going to become increasingly prevalent in the coming years.

“My own view is that augmented reality is the larger of the two, probably by far, because this gives the capability for both of us to sit and be very present talking to each other, but also have other things visually for both of us to see - Tim Cook.”

Despite the recent surge in popularity and hype around AR, resources for development in this area are somewhat scarce and lagging behind. While there exists a number of authoring tools aimed at making game development more accessible to non-programmers, making an AR application in particular still requires more than a basic understanding of both programming and AR components.

Another field experiencing rapid growth in recent years is the board game industry. In 2012 The Guardian stated it was “A Golden Age for Board Games”, stating board games have seen a growth rate of 25% to 40% year after year [6]. There is an increasing number of indie board game creators and hobbyists pushing the boundaries of traditional board games and some are already starting to incorporate AR elements into their games. HoloGrid: Monster Battle [14] is an AR board game launched this year after a successful Kickstarter campaign. AR has the potential to bring otherwise static board games to life by adding a new dimension to the game play, enhancing visuals and a new method of input. Currently however the majority of board game creators do not have software development experience, so in order to take advantage of AR must hire a developer, which is not always viable, especially for a small business or hobbyist.

The purpose of this paper is to attain whether the development of new set of resources and tools, specifically designed to aid in the creation of an augmented reality board game for those without a background in software development, would be beneficial to current and prospective board game creators. Augmented reality and boardgames are two industries on an upwards trajectory individually but there seems to be a disconnect between the two meaning that there is currently significant enough obstacles preventing board game creators from implementing AR. Perhaps if there existed a toolkit specifically aimed at overcoming these obstacles, far more AR elements will start to appear in boardgames.

2. CHALLENGES WITH AR

Augmented reality systems are measured by how realistically they augment digital information into the physical world and despite recent advancements there still exists several challenges and issues with AR technology. There doesn't exist an AR system in the world that can consistently augment virtual objects into physical space in real time that are indistinguishable to a human eye. The human brain is fine tuned to notice imperfections in an environment and are immediately drawn to something that is even slightly out of alignment, has the wrong scale or is poorly rendered. Using current technology, nearly all the steps in augmenting an object are imperfect, making it very difficult to provide realistic augmentation. Current AR systems aim at being realistic enough for a user to be able to ignore these imperfections.

Firstly sensor data, primarily the camera, is never flawless. Camera noise is impossible to avoid, poor lighting or motion blur can make visual trackers difficult to track and gain information from. Even without noise gaining meaningful information about an object's 3D location, rotation and scale from a 2D image is difficult even for an advanced tracking system, especially if the object is moving. The unreliability of AR means that there are difficulties when a game's mechanics system rely on information obtained using image tracking. An erroneous reading has the potential to propagate through the game and this should be handled. Data from other sensors, including the accelerometer, can be combined with camera data so to reduce the effect of camera error. A smartphone's accelerometer can detect minuscule movements to a device and adjust the imposed location of the virtual object accordingly.

The processing power of devices is also a bottleneck, to process and track several moving visual markers at real time, all whilst superimposing and rendering dynamic 3D Objects and running complex game mechanics takes an enormous amount of processing power. Even the latest smartphones can have trouble keeping up, the mainstream AR SDK’s all impose limits on the amount of simultaneously tracked images. This is a major limiter for a game designer.

3. AR ON MOBILE

Overtime the average smartphone has become powerful enough to process AR in real time, there has been an influx of mobile AR applications for mobile devices.. A large percentage of the population already carries a device which is light, portable and comes built with a high resolution display, high resolution camera, a wide array of sensors and ample computational power. Less prevalent are Head Mounted Displays (HMD) like Google Glass. HMDs can be uncomfortable and can cause both physical and psychological issues for the user, for example HMDs have been known to cause immersion injuries and physical discomfort [5].

Mobile AR games commonly use Visual Markers [4][8][12], these are visual clues that can be easily recognised by the image detection system. Once a marker is detected, the position, scale and rotation is calculated using the perspective of the marker and then transferred into virtual information. As the system relies on knowing where these markers are at all times game design can be somewhat limited. Due to the amount of sensors on a smartphone, it is possible to collect more information about the surrounding world which could not be achieved on the camera alone. The phone can also act as an input device to manipulate digital data. Due to the physical nature of a board game much of a smartphone's sensor data will not be required. Board Games typically involve one or more people sitting equal distances around a centralised board. A board game specific toolkit should be selective with the functionality it offers, board games do not need access to features such as GPS data, so to include this in the toolkit would only complicate the interface further.

4. RELATED WORK

Several authoring tools exist for building AR applications, these can be broadly categorised into two types; tools designed for software developers and tools designed for hobbyists and non-programmers. In the former, the tools are typically complex code libraries which require programming knowledge to author the application, but allow for a much finer degree of control. Tools which are not aimed at software developers require layers of abstraction to be added and lower level programming capabilities to be removed or hidden. Applications like these include a GUI for building software whilst minimising the need for the user to write code.

4.1 AR Authoring Tools

Recently there has been a number of AR dedicated SDK’s released to simplify the process. Some of the most popular are Vuforia, ARToolKit and ARLab. Essentially these provide an interface for whichever engine or programming environment is needed so to trivialise dealing with the more complex AR aspects. For example developers no longer have to worry about creating their own image tracking system as each SDK already provides this functionality. Each SDK has its own advantages and disadvantages but all provide an array of useful functionality for developers. The release of these toolkits has made AR a lot more accessible to hobbyist developers but they are not yet at the simplicity level so that a creator with no software development experience could pick up with ease. Vuforia’s SDK works in Visual Studio, XCode and Android Studio, all of which require a more than basic understanding of computers and programming to navigate confidently [1].

The Designers AR Toolkit (DART) [10], built on Adobe Director, was developed for rapid prototyping of AR applications by designers. It was designed specifically to address problems that designers faced whilst working with AR. Whilst DART is just a prototyping tool and couldn't actually author AR applications it is an excellent example of using software abstraction to simplify more complex ideas so that non-software developers, in this case designers, can use their time designing rather than struggling with programming. This is something that any toolkit for creating AR board games should hope to replicate. Board game creators should expend their efforts on the design and concept of their game, and not with programming complex AR concepts. One of DARTs better features was its ability for its users to modify its functionality to suit each individual user's needs using a scripting language. However for board game creation this may in fact be more of a hinderance, one of the things currently deterring creators from implementing AR is the perceived learning curve when it comes to learning to program to a necessary level. A cleaner more intuitive UI would be more welcoming to the majority of creators.

AR SPOT [13] is an augmented reality authoring environment for children. It is an extension to MIT’s Scratch project and provides a number of AR related tools in a colourful and intuitive way. Scratch is used as an introductory programming language in schools and allows children to easily create small programs, games or animations by connecting blocks, which are pre-made scripts, to form logical statements. AR SPOT extends the standard scratch library with a camera view and a collection of AR specific blocks. Similar to DART, AR SPOT simplifies the task of implementing AR into an application by abstracting many of its complexities. AR SPOT excels at teaching people the basics of AR programming, Scratch itself is limited however, and if a user has the intention of releasing their board game it would need to be created in other languages. Scratch, however simple, is still a programming language so an understanding of basic programming constructs is still required. A toolkit aimed solely at providing functionality for AR board games could abstract even further and provide functionality for very specific tasks, such as creation of a board or a counter.

Auragame [7] was created to demonstrate the potential of mobile AR authoring tools. It challenges the players to solve problems and enigmas by gathering AR objects placed around the environment. The game was created without code using the platform Aurasma. The creators of the platform claim AR games can be created with little or no programming knowledge. There were some slight limitations to Auragame, issues viewing the physical objects even when positioned correctly hindered how a person could play the game as they could not see the puzzles they were trying to solve. This Aurasma platform was a somewhat successful attempt at creating an authoring tool that requires no software development skills, outlining the difficulty in striking the correct balance between abstraction and flexibility. If the platform is too simple, then creators will not have the control to create what they need.

4.2 AR board games

There have been numerous attempts at incorporating AR elements into board games. The transformation of the board game Monopoly into an Augmented reality game using a webcam is shown in [11]. The creators use a webcam to detect visual markers onto which each game element is superimposed, there markers are simple but distinctive QR style codes. The board and each piece had its own marker and thus can move independently of each other. The game is then simply played like standard monopoly, the game's mechanics or rules are not altered by AR. The result of the added AR is a far more interesting and dynamic looking board, the digital objects are animated and appear to move. It would be possible to augment nearly any board game in this way as the game itself remains unchanged. A toolkit to aid board game creators could make it extremely easy to set up visual markers and animate traditionally static games.

On the other hand, Hologrid: MonsterBattle [14] is built from the ground up with AR. HoloGrid is a turn based strategy card game that has AR elements built into the gameplay mechanics. Each player uses their mobile device with which they fix pointing towards the virtual game play area. Tactical decisions and choices are made either by playing physical game cards, which double as visual markers for AR, to the game play area or by using the touch input on their device. The game play is fully animated and the scoring and game mechanics are handled digitally by the smartphones. This is a step beyond the Monopoly example for a few key reasons. Firstly aspects of the game rely on augmenting varying amounts of information, such as strength or location of the opponent, to each player for them to make their decisions. In a traditional board game the state of the play is largely visible to all players, AR significantly extends the possibilities for game mechanics. Secondly the sheer amount of dynamically changing information stored and calculated by the application is well beyond anything that could reasonably integrated into a traditional board game. Each game character has their own set of stats that can are influenced by playing the game in different ways. The game flow and rule system is also implemented using the application, this can effectively eliminate any cheating or ambiguity over how to play and reduces the time and effort needed to learn the rules. Finally the touch screen on mobile devices allowed the developers to provide for more varied and precise control over each choice.

Battleboard 3D [15] is an interesting concept somewhere between AR Monopoly and Hologrid that places visual markers on Lego. The game is played out solely in the physical world, however connecting Lego bricks together causes a fight sequence between the two to be initiated. The outcome of each battle is calculated digitally using the application. Like in the Monopoly example AR serves only to visually enhance the game and the state of the board is the same to all users.

The primary advantage of incorporating AR into any board game is the huge visual enhancement possibilities, however HoloGrid and to a lesser extent Battleboard3D show that it can open up possibilities for new mechanics and game play to developers. The potential complexity of games created using digital elements is far superior to what is possible in traditional board games. This is because the players are no longer relied upon to carry out the game's mechanics, allowing more complex systems. The extent to which a game uses AR can vary from animating a select few game components all the way to something as complex as Hologrid. Game designers must decide how much of a game exists physically and how much virtually. An AR game consisting entirely of virtual game objects would need a single visual marker and some sort of digital input. With the advent of mainstream AR the line between board games and video games is blurring.Turn based strategy games, in which each player is provided with information about the state of play in order to make decisions, lend themselves to AR. In the future we could see even more complex games similar to Civilisation V created as a table top AR game.

4.4 Conclusion

To summarise, the board game industry could clearly benefit from further AR integration. HoloGrids hugely successful Kickstarter campaign shows that there is consumer interest in AR board games. One of the reasons for this seems to be the difficulty and accessibility of developing for AR, specifically for board game creators, who tend not to have software development experience. All the existing SDK's, whilst simplifying the process of AR development, are aimed at current software developers. Knowledge of development complex development environments like Xcode and Android Studio is still needed. Smaller projects like SPOT AR provide an even simpler UI but is really targeted at teaching children core AR principles and doesn't provide the functionality to create commercial applications.

The issue for board game creators is that they are currently provided with too much functionality making the learning curve too steep. If there existed an SDK, designed at solving very specific board game creation related tasks, the time to learn the new system would be reduced greatly. As discussed, the extent to which board games use AR is on a scale. At its most basic level the SDK could provide a clean, quick and intuitive way of assigning visual markers to virtual 3D objects. This would allow a creator to make many games similar to the AR Monopoly rapidly. The complexities of AR would be hidden from the user, they would simply assign a 3D Model to a visual marker. The mechanics in a game like this would be enforced by the player.

To start to implement a rule set that will be handled by an application, the SDK would need an understanding of the board game mechanics. Most existing board games share many of the same properties and can be described in terms of a board, various game objects and a rule set. The SDK could provide the ability to create and configure each set of properties then use this information to make the game. This would allow the creation of more complex games, where the games mechanics are handled digitally, like Battleboard3D. It would be important to give enough control to the creator without over complicating the interface.

5. A SYSTEM TO ENABLE AR BOARD GAME AUTHORING

In order to test the viability and usefulness of an AR Board Game creation toolkit, we are going to develop and use the created application to find out how participants interact with it compared to other systems. The toolkits primary goal is to provide an interface so that individuals, without a programming background, can create a variety of AR enabled board games. A toolkit of this nature could be taken far beyond the scope of this paper, but for the purpose of assessing a toolkits potential usefulness, a select subset of tools that are most important for creators will be implemented. This iteration of the toolkit is geared towards grid based games like draughts and chess.

5.1 Approach

It was decided that the toolkit will be created in the form of a custom editor for the Unity gaming engine. A number of factors went into this decision. Custom editor windows can be used to extend the functionality that Unity offers and can float free or docked like the native unity windows. The toolkit will be designed to be tabbed next to the Unity inspector. Ideally the toolkit would have been a standalone application independent of Unity so as to only offer tools specifically relevant to board games, but building on top of Unity brings certain advantages. Unity has built in support for building to all major mobile platforms, allows us to use the Vuforia SDK to build AR related tools and finally Unity's built in graphics engine is not something we could have replicated.

It is vital that the interface is intuitive and self explanatory, visual clues and previews will update in real time when settings are changed. We envisage that a high percentage of potential users will solely be assigning 3D Models to visual markers so this should be prominent, the complexities of image recognition and tracking should be hidden from the user. Letting the user define a logic system for the game is more open ended, we want to enable users to create as many variations as possible without bombarding them with too many options.

On completion, we will assess the toolkit in two ways. First we will directly compare completing specific tasks with and without the toolkit. Examples of these tasks might be creating a board or superimposing a virtual counter on a visual marker. We will compare time taken, flexibility and simplicity. Secondly we will invite participants for user testing. Ideally participants will not have a software development background. During the user testing phase we don't want to give participants too much direction as we want to assess how each person interacts with the software as if they had downloaded it from the Asset Store. We may give each partaker an open ended task and assess how they go about completing it. This is where we hope to find out which aspects of the toolkit are beneficial, whether people found certain aspects limiting and how it could be improved in a second generation. Hopefully we can use these finding to form conclusion as to whether an AR board game toolkit would be advantageous commercially.

5.2 Board Game components

For the toolkit to provide functionality beyond just animating board game elements we need to be able to express all board games in terms of sets of certain elements. In order to implement a logic system that controls the game's mechanics it is going to need a conceptual understanding that various elements are not just 3D Objects but have different properties and ownership.

5.2.1 Squares and Boards

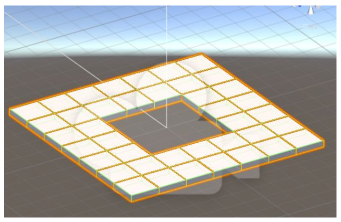


Figure 1. Shape and size of board

The vast majority of board games have a board on which the game takes place. A board will be made up of collections of individual squares. This allows boards to be varying size and shape depending on the game being created. Each square contains an ID that corresponds to its position in the board.

5.2.2 *Pieces and Teams*

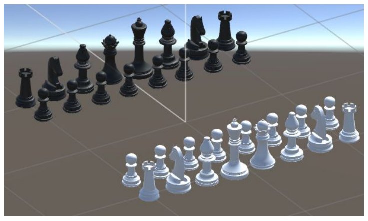


Figure 2. Pieces

Pieces describe the counters or characters of a board game, each game might have many pieces. Chess, for example, has six different piece types and 32 piece instances at the start of play. Each piece belongs to a team, conceptually each team corresponds to a human player. Teams contain references to all of the pieces in their possession. Pieces contain references to the square on the board that they are located and a list of moves that it is permitted to make.

5.2.3 *Moves*

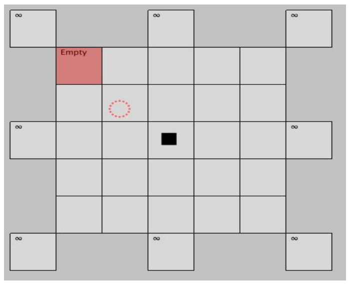


Figure 3. Permitted Move

Each piece contains a list of moves in which that particular piece type is permitted to make according to the rules of the game. A move consists of a destination square, this is not an absolute square on the board but a relation from a pieces current square. For example, one of the permitted moves for a knight in chess would have the destination square two up and one left from its current square, along with separate moves for the other seven possible moves. Moves also contain conditions for empty and opponent squares. These must be satisfied for that move to be offered to the player. For example, a diagonal capture move in draughts is only permitted if both the immediate diagonal square contains an opponent piece and the square diagonally after is empty. Each move stores all of its own condition squares. Moves are associated with a type of piece, all instances of that piece type will have the same list of moves.

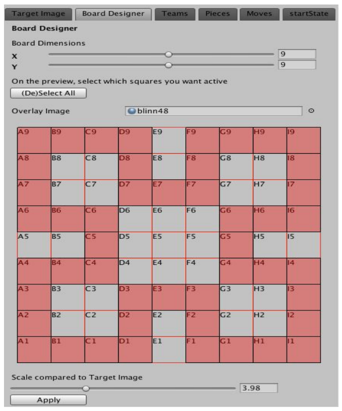
5.3 System Design

A tabbed UI was chosen for the toolkit as it allows users to navigate between the various tools that it offers. Each board game doesn't necessarily have to be created in any particular order so users can alternate between the available tools and tinker with settings. Additionally if a user only needs a select few features, they can navigate straight to the tabs of their choosing.

There is a tab for configuring the main target image, configuring board design, managing teams and pieces, managing moves and setting up the start state of the game.

The main target image is for the visual marker that the board and therefore game mechanics will be attached to, the majority of users will use this tool so it is presented first. Pieces that do not use their own visual marker will use the main target image. At any point whilst using the toolkit it is possible to run the game in unity or even build to android or iOS. The game is automatically given a simple UI and input system.

5.3.1 *Board Designer*


Target Image
Board Designer
Teams
Pieces
Moves
startState
(De)Select All
A9 B9 C9 D9 E9 F9 G9 H9 I9
A8 B8 C8 D8 E8 F8 G8 H8 I8
A7 B7 C7 D7 E7 F7 G7 H7 I7
A6 B6 C6 D6 E6 F6 G6 H6 I6
A5 B5 C5 D5 E5 F5 G5 H5 I5
A4 B4 C4 D4 E4 F4 G4 H4 I4
A3 B3 C3 D3 E3 F3 G3 H3 I3
A2 B2 C2 D2 E2 F2 G2 H2 I2
A1 B1 C1 D1 E1 F1 G1 H1 I1
Apply
Board Designer

Board Dimensions

X 9

Y 9

On the preview, select which squares you want active

Overlay Image blinn48

Scale compared to Target Image 3.98

Figure 4. Board Designer

The actual board designs vary significantly from game to game, if digital game mechanics are required it is important that creators can accurately describe to the system their chosen board design. Creators specify the master dimensions and then use the clickable preview to select precisely the shape of the board, allowing just about any shape consisting of multiple squares. Users specify the artwork for their board, although this is part of a board game creators job so a specialised tool would have been redundant. The artwork field accepts most image files and automatically scales and resizes to the specified board. Upon completion the toolkit builds the specified board in the Unity scene view.

5.3.2 *Teams and Pieces Tab*

The Team and Piece tabs are for adding, deleting and managing each games teams and piece types. The entire window is dedicated to these tabs as more complicated games might have several of each. The system relies on these constructs in order to implement game mechanics so it's necessary that they are easily described correctly by the creator. This is where the 3D Model is specified for each piece. Teams are assigned a unique colour and piece types are assigned a unique symbol. These identifiers are for use within the toolkit only. whenever aspects of a team or its pieces are displayed in the toolkit, they are recognisable by their colour and instances of a piece type are recognisable by their symbol.

5.3.3 Adding Moves
It was crucial that each pieces moves can be accurately described visually. A piece might have several potential moves and there might be several pieces so speed of input was a factor here. A subsection of the board is displayed to the user on which they must define a moves characteristics. As described previously moves consists of a destination, empty square conditions and opponent square conditions. User can click on the dynamic preview and specify each of the moves three characteristics. There can be multiple condition squares, but only one destination.

5.3.4 Game Setup and Start State
Once the piece types and teams have been configured the game setup can be described. This is where creators add each piece to either the board or their own visual marker and ultimately to the Unity project. Users have two choices here, they can view this page in piece view, board view or flick between the two. Piece view gives the user finer control over each piece added, they can specify a piece's team, size, name and then choose whether to attach it to its own visual marker or to a particular square on the board. In Board view it is possible to rapidly add many pieces to either team by clicking on the dynamic preview of the board. This is handy if a game, like draughts, involves lots of identical or similar pieces.

6. TESTING AND FEEDBACK

6.1 Comparison testing
After creating the Toolkit authoring tool, we decided to evaluate how the toolkit performs on particular tasks compared to performing the same task without the tool kit. We want to assess the flexibility, speed and simplicity of constructing certain board game elements. Realistically, the toolkit has an unfair advantage at this stage of testing as it is known to and was designed specifically to handle the creation of these game elements. But it will be valuable to gauge whether or not it actually offers an advantage and by how much. For the comparison we will use the standard Unity editor tools, Unity already does a great job at abstracting many complexities for developers, particularly when handling 3D graphics.

The first task is to create a board in an empty Unity project. We define a board in our system as a collection of squares with an an image of the board design placed on top.

Table 1. Comparison between tasks performed with and without the toolkit

	Speed	**Flexibility**	**Simplicity**
1x1 board	Editor	N/A	Same
8x8 board	Toolkit	N/A	Toolkit
Ludo style board	Toolkit	Editor	Toolkit

The one by one board, consists of a single 3D shape with a shader applied to it. There is no point opening the toolkit for such a straightforward task. As soon as multiple squares are required it takes substantial time to move and align each square, a separate plane also has to be added to display the image. Using the toolkit, boards of any size and shape can be created in a matter of clicks, whereas time taken increases with each square added when using the editor. Developers would look to do this procedurally but this is not an option for non programmers. Technically you have slightly more flexibility with the editor as you are not bound to a grid.

The next task was to assign a virtual 3D object to a visual marker. We will assume that both object and marker were in the project folder as this is the same for both.

Assign Marker	Same	Editor	Toolkit

A person who has knowledge of the Vuforia SDK will be able to do this quickly without the Toolkit, it is a case of dragging an image tracker and a 3D object into the project hierarchy and then configuring the image in the inspector. However, for those that don't, it is not an intuitive process. Vuforia provides a large amount of useful game objects but to identify and locate the correct one would be time consuming. In terms of flexibility, the built in inspector offers far more than just the size scaler that is offered in the toolkit, most of which are redundant except in exceptional circumstances.

The final task is to make a digital counter move from one square to another. This is difficult to measure as this is fairly built into the game mechanics implemented by the toolkit. But let's suppose all the editor has to do is move a counter from point A to point B in a smooth manner, on a button press.

Move A to B	Toolkit	Editor / N/A	Toolkit

The toolkit includes a function that allows for smooth following between two squares on the board. This is all handled behind the scenes, the creator simply specifies the moves that each counter is allowed to do. The total number of squares specifiable by the current toolkit is limited, so a bespoke solution created via script would offer more flexibility both in terms of where the piece goes and how it gets there. Although, without finding and importing a third party asset or script, this is not currently possible without some basic programming.

What is immediately apparent from gathering these results and using the toolkit is that carrying out a task in the Editor rather than the toolkit will almost always result in more flexibility and configurability to the task. This is hardly surprising as the toolkit is essentially providing an abstraction of what the editor can offer in the hope that it can make certain aspects more useful to the user. Naturally, the toolkit will never be more capable of more than the editor but it must provide enough features for a board game creators needs. These test results suggest that the feature set has not yet reached that level, currently the the toolkit can do specific tasks a lot quicker and more efficiently but cannot yet replace the editor altogether. In its current form it could provide a very useful helper tool that could do a large percentage of the AR board game creation and actually simplify and speed up development significantly.

6.2 User Testing

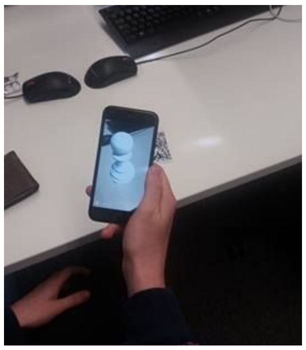


Figure 5. Chess piece augmented on mobile device

A user test was then conducted with six students from the University of Nottingham, aged from eighteen to twentyfive. We specifically sought out participants with little or no prior programming experience. Each was provided with a blank Unity project with the toolkit tabbed to the left of the screen, a selection of 3D models, QR Codes for visual markers and basic board patterns were also left within the project folder. One of the aspects we wanted to test was how intuitive the software was to someone who is playing around for the first time. Three participants were given an instruction sheet describing the basic functionality of each tab, the remaining three were left to their own devices. For both groups we made sure they understood the basic principles of AR, provided some basic features of the unity editor and gave assistance when asked, but otherwise let them figure it out themselves. The testers were encouraged to express their thought processes verbally while performing the tasks.

After letting each of them play around with the toolkit for a while we challenged them to create a chess like board game. This is an open ended task and there are a number of ways one could go about completing it.

6.2.1 Summary of user testing.

The first thing to note was that the sample group that were provided with instructions were not noticeably quicker to get to grips with the toolkit. For all participants it took a few minutes of clicking around, making a mess and experimenting with different features before anybody managed to create anything meaningful. All participants successfully managed to create the main target image and add some sort of board to the scene. All but one participant without an instruction sheet and all with an instruction sheet grasped the concept of the board designer, allowing them to create the exact shaped board that they desire. However, there was a tendency to spend long times creating ridiculously complex boards. All of the participants natural instinct was to create a new visual marker for each piece that they added to the scene. With time, all but one participant managing to place pieces onto specific squares of their choosing. Only two out of the six got any sort of move to work on a placed piece without assistance, both had an instruction sheet. There was some confusion over some terminology, using "teams" to describe each human player, this had to be explained or at least clarified to everyone.

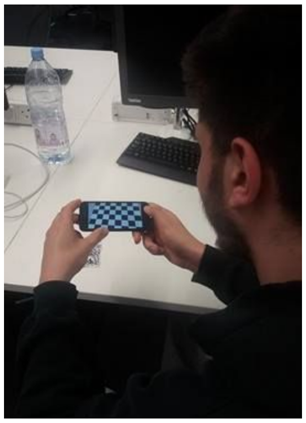


Figure 6. Board augmented on mobile device

When challenged to create chess, all but one candidate picked the chess board artwork out of the assets folder, created an eight by eight board and added at least some of the chess 3D Models to either the board or a visual marker. Two participants fully understood the system and created two teams, seven piece types and started implementing each pieces move set with relative accuracy. Both were confident that with more time they could have created a semi working chess game. However, one of the two, commented that they thought the toolkit was specifically geared towards created games like chess and struggled to envisage creating another board game type, such as monopoly. One other participant began assigning unique visual markers to each type of piece, this would have also resulted in a playable board game eventually.

6.2.2 Analysis of test results

In hindsight, it would have been preferable to test the toolkit with subjects that had more of an insight into board game design. We were focussed on testing using users that didn't have any programming experience but the average user of the proposed toolkit would have a far greater insight into the structure of how board games are put together than a random selection of people. Contributors of this nature were hard to come by but perhaps would stretch the limitations of the system slightly further. This is perhaps why all candidates tended to try and add a visual marker to each game piece. Someone with an actual AR game design worked out, would probably have behaved differently.

Despite the shortcoming of the experiment the interaction by most of the participants was promising. None of the six had set eyes on the program before the experiment and all managed to create some, if not all, key elements of an AR Board Game in less than an hour. If you consider that none of them have a programming background and only two are studying, or have studied, a STEM subject it would have been inconceivable for them to create what they did on existing commercial AR authoring tools in such a short amount of time. Subjects found it straightforward to assign 3D models to visual marker, this is an important victory for the tool kit. As mentioned previously, an abundance of new and existing games could implement basic AR animation using just this feature. This feature alone would be useful to many creators.

However, there are some important negatives to take from this study, it was clear that the way in which a user is encouraged to define a game into the system is more familiar to those who have at least come across programming constructs such object orientated programming. A slight re-design would be needed to make the environment structure even more recognisable conceptually to non programmers. This would include re-imagining some of the terminology used to describe some aspects of the game. It also clear that despite our best efforts to keep the UI intuitive and self explanatory, documentation and instruction is needed, especially if the toolkit is inevitably going to be expanded.

One of the participants mentioned that they felt the toolkit was geared towards specific types of games. This is certainly somewhat true at this point but the way in which the system describes games has room to and could easily be expanded, for example, with more complex and varied move types or an easier way of expressing non player game items. The toolkit was created to test the potential usefulness of a an AR board game authoring tool, if found to be so its feature set can be expanded to support the creation of a far more varied set of board games. This study highlights the fact that with some minor upgrades the toolkit can be very quick, intuitive and efficient at creating what it is capable of creating. The challenge for future generations of this toolkit is to be capable of creating everything board game creators need to be created. This study highlights some limitations in the systems rule definition input.

7. DISCUSSION

The authoring tool that we have created, aims at providing an extra layer of abstraction, on top of what traditional SDK's like ARToolkit and Vuforia are currently offering, so that board game creators can implement varying levels of AR integration into their games. One issue with these systems is that it was too difficult for non-programmers to learn how to use them in a reasonable period of time. Our toolkit certainly goes someway to addressing this, all participants that it was tested on managed to create some sort of AR content within several minutes of picking up the software. This is essentially achieved by trying to prioritising and display only the features that board game creators are likely to need. With less unnecessary options, the system is more straightforward and quicker to pick up. The issue with hiding options and tools from the user is that you simultaneously reduce how much control the user has over the system and therefore diminish the range of things that it is possible to create. The key is to abstract to a level so that your target user can use and understand the system without limiting development possibilities. The authoring tool we created goes too far and some participants noted that they felt limited in the type of game they were able to create. By extending the feature set to rectify this, we increase the learning curve to learn the system again. Perhaps this is why all the major AR toolkits abstract concepts to a similar level. To reduce the feature set by any more starts to limit new applications.

In the tests, participants had a tendency to create a new visual marker for every new piece that they created, this is the simplest way of extending a board game with AR. Maybe this suggests that AR games like AR Monopoly are what lend themselves to using a toolkit of this nature. Games like this are played out in the physical world but animated and augmented visually with AR. This is simply a case of assigning multiple visual markers to 3D Objects, something that our toolkit excels at. Practically all existing board games could implement AR in this way. Our comparison testing suggested that our toolkit is a better system than the unity editor at implementing this particular feature. This is an example of where automating and simplifying aspects of one of Vuforia's features has not limited the creation potential for our target user. All other features in the toolkit should aim for the same.

Throughout the user trials, even with our toolkits abstract design, it was noted repeatedly that certain aspects would have made more sense to someone with a development background, who was capable of conceptualising how elements of the game is defined and stored. One area where this issue manifested was the move creation tab. To try and combat this issue in future generations of the toolkit there could be two distinct approaches. To continue trying to hide and present this information in a format more understandable to non-programmers, which could further complicate the system. Alternatively a change in philosophy could be explored where the logic system should be mad really upfront and transparent to the user. In SPOT AR, logical connectives and AR components are presented in a way that is supposed to be understood by children. If these concepts can be defined in such an intuitive way in AR SPOT, then it should be possible to replicate the same sort of system in an adult AR authoring tool. Scratch, is a full scripting language and if understood correctly would allow for far more complex game mechanics to be developed. This would in turn allow for far more complex games to be defined by the user as it could also be used to alter all aspects of the game's mechanics. For example, the game could store user defined information and have it dynamically change according to game play events.

8. IMPLICATIONS FOR DESIGN

The primary challenge in making a second generation of our toolkit would be to oversee the expansion of the toolkits feature set without over cluttering and losing how easy the system is to learn. Clearly, the set of games that can be created by the toolkit needs to be enlarged, this can only be done by giving more tools and displaying more options to the user.

Ultimately, in order to be able to create more complex games, each games mechanic need to be able to be defined or at least altered by the creator. To do this, a greatly improved logic system should be implemented. As previously discussed, an interface similar to AR SPOT could be explored in the hope that it would provide the user creative power to define more advanced systems without having too steep learning curve. A better defined logic system could accept more inputs from the sensors on mobile device, implement a scoring system or vary the amount of information that can be displayed to each human players AR view. If an interface such as this is deemed to be unfeasible, the toolkit could easily be scaled back to offer a useful helper tool that offered board, piece and visual marker creation.

Another feature that should be implemented is support for game items that are not pieces, like dice or cards. This will massively increase the range of games that can be defined. These new game items will need to be conceptualised and given a mechanism to be described in terms of the system mechanics. Even without a revamped logic system, a way for creators to better specify digital inputs would be beneficial, as the drop down inputs included in this toolkit are not suited to all game types.

9. CONCLUSION

Ultimately, it can be concluded that an authoring tool developed specifically for non-programmers to create AR board games can be hugely beneficial in simplifying and speeding up certain aspects of their development.

However, when it comes to describing and defining a game's mechanics, it is not feasible to create an abstract authoring tool capable of providing enough configuration to describe all possible AR board games without using the configurability that a full scripting language offers. A GUI offering the functionality of such a language would be extremely large and complex and learning such a system would be more difficult than learning the original scripting language. Using a Scratch like language, if explored, could potentially offer a halfway house that lets users create comparably basic logical structures to describe their games mechanics.

A toolkit that runs alongside a larger development environment, such as Unity, to provide support for the creation of some aspects of an AR board game is the most viable option. The toolkit can be used to quickly and accurately describe aspects that are common in all games, like boards, pieces and the visual markers to track them. But for more complex ideas and additions the user can fall back on, in the case of Unity, C#. This is unfortunately not ideal for board game developers with little or no programming knowledge.

10. ACKNOWLEDGMENTS

We would like to thank Dr. Martin Flintham for overseeing the project. Assistance was always available to us had we needed it.

11. REFERENCES

[1] Amin, D. and Govilkar, S. 2015. Comparative study of Augmented Reality SDKs

[2] Avram, A. 2017. IDC Study: How Many Software Developers Are Out There?. *InfoQ*.

[3] Azuma, R.T., Baillot, Y., Behringer, R., Feiner, S., Julier, S., and McIntyre, B. 2001. Recent Advances in AR. *IEEE Computer Graphics and Applications, 21* (6). 34-47

[4] Cooper, N., Keatley, A., Dahlquist, M., Mann, S., Slay, H., Zucco, J., Smith, R., and Thomas, B. H. 2004. Augmented Reality Chinese Checkers. In *International Conference on Advances in Computer Entertainment Technology*.

[5] Costello, P.J. 1997. Health and Safety Issues associated with Virtual Reality - A Review of Current Literature,

[6] Duffy, O. 2014. Board games' golden age: sociable, brilliant and driven by the internet. *The Guardian*.

[7] Fernando Maia, L., Rodrigues, W., Viana, W., and Trinta, F. 2016. Auragame: A Case Study of a Zero Programming Augmented Reality Game SBC – Proceedings of SBGames 2016

[8] Govil, A., You, A., and Neumann, U. 2000. A Video-Based Augmented Reality Golf Simulator. In *International Conference on Multimedia*.

[9] Lofgren, K. BigFishGames. 2016. *2016 Video Game Statistics & Trends Who's Playing What & Why*.

[10] Macintyre, B., Gandy, M., Dow S., and Bolter J, D. 2004. DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences. *In: Proc. of the ACM symposium on User Interface Software and Technology* 6(2), pp 197–206

[11] Molla, E., and Lepetit, V. 2010. Augmented Reality for Board Games. *IEEE International Symposium on Mixed and Augmented Reality*.

[12] Oda, O., Lister, L., White, S., and Feiner, S. 2008. Developing an Augmented Reality Racing Game. In *INTETAIN*.

[13] Radu, I. and MacIntyre, B. 2010. AR SPOT: Authoring Augmented-Reality Experiences through Scratch. Research Paper presented at Scratch MIT Conference. Boston, USA

[14] Tippett, P. 2016. HoloGrid Monster Battle.

[15]Troels, L., Andersen, T., Kristensen, S., Bjørn, W., Nielsen, and Grønbæk, K. 2004. Designing Augmented Reality Board Games: The BattleBoard 3D experience. In *Conference on Interaction Design and Children*.