Multiplayer Game Design: Performance Enhancement with Employment of Novel Technology

B. M. Monjurul Alom
Assessment Research Centre
Melbourne Graduate School of Education
The University of Melbourne

Claire Scoular
Assessment Research Centre
Melbourne Graduate School of Education
The University of Melbourne

Nafisa Awwal
Assessment Research Centre
Melbourne Graduate School of Education
The University of Melbourne

ABSTRACT

Online multiplayer educational game is designed to support collaboration and assess different cognitive and social abilities among students. The educational games are developed to capture student responses or actions, both shared and unshared, within the game environment and extrinsic resources. This paper describes the implementation of new technology with better performance over existing HTML5 and Flash platform to develop such games, which have been developed as part of the ATC21S™ research study by the University of Melbourne. CONSTRUCT2 has been used in preference to other available technologies in creating the games to deliver a reliable experience for students across all browsers, platforms, and devices. The multiplayer component of the games was maintained with the use of the AJAX and Web Socket application which allowed the communication protocol between the client and server to be established. Embedded Canvas on CONSTRUCT2 is used to create all animations and game objects. The paper will describe the issue such as multiplayer game design with new technology and optimization of collaborative game design using dynamic platform structure called 'CONSTRUCT2' over HTML5 to facilitate the game flow and a positive user experience.

General Terms

Semantic Web Technologies and Social Semantic Web, Applications of Computer Science, Information Personalization.

Keywords

CONSTRUCT2, HTML5, Canvas, Web socket, Game design, AJAX, Collaborative Problem Solving, Web Graphics Library.

1. INTRODUCTION

The games discussed in this paper are developed by CONSTRUCT2 platform that were initially designed by HTML5, as part of the Assessment and Teaching of 21st Century Skills (ATC21S™) study presented by (Alom et al. 2015). CONSTRUCT2 is an HTML5 based two dimensional (2D) game editor which allows quick creation of games in a drag-and-drop fashion using a graphical editor and a behaviour-based logic system. The games emphasis on collaborative problem solving; some of games were initially developed by Zoanetti (2010) for single student use in computer based problem solving. The expansion of online games for education has advanced quickly in recent years. The need for digital games that can be reformed to permit students to play and express themselves during their learning is presented by Marchetti and Valente (2014). For example, the strategy and combination of the application SimApps within GameScapes is elaborated by Amresh, Clarke and Beckwith (2014) and describes a novel method of integrating rich narrative based storytelling within short repetitive game play procedure to inspire student learning and improve preservation. To incorporate multiplayer use Masuch and Rueger's (2005) paper has described on teaching collaborative game design within a precise platform referred to as OpenCroquet. A common framework of the game design process, the structure of digital games and the requirements for the use of OpenCroquet as a platform for collaborative game design is described.

Presently web based games developed by developers are very interactive and for this developers need to include animations, video stream, music and other forms of multimedia into web based games. There have been limited tools available for use, such as Flash or Silverlight, Flex or JavaScript up to now. Nevertheless, these preferences can take a lot of time to integrate and increase the difficulty of development. With the arrival of HTML5 it is promising to embed video and audio, high quality drawings, charts and animation and other rich content without using any plugins and third party programs as the functionality is constructed into the browser. To support interactive dynamic bitmapping, drawing, animation, and simulations canvas which is an element of HTML5, has been used. The Canvas element allows for an easy transition from plain to dynamic web pages that are applicable to mobile devices, perhaps why it is popular with developers. The use of HTML5 with Canvas for game development with user impression is described in detail by Crockett's (2014). This paper has focused the advantages of HTML5 over using Flash applications.

Due to the benefits outlined above HTML5 was the application of choice for development of online educational games for the ATC21S™ study is presented by (Alom, Awwal, & Scoular, 2015), in order to provide a consistent experience for students across all browsers, platforms, and devices. Multiplayer games were designed in this study, allowing two students to cooperate by sharing resources and communicate through an embedded chat box. To assess the skills of students, this facility can be used through their responses or actions within the game environment. Networking technology can be used by multiplayer games to allow game play from dispersed locations, and even over a greater distance. The multiplayer component of the games is supported by HTML5 and Web Socket, which dictates the full communication network that operates through a single socket over the Web.

There were some complications to develop these games with HTML5 (Alom et al. 2015). Specifically, two libraries such as Phaser and KineticJS were used to develop these games in the HTML5 platform. This resulted in very slow interactive drag performance while running the game developed by the library KineticJS. To establish and maintain socket communication between the server and client was very challenging via the HTML5 platform. The programming code was also prolonged using Phaser library.

To overcome some of these problems, the redevelopment of games in CONSTRUCT2 are described in this paper using the method of 'event sheets'. Each event sheet has a list of events, which comprise conditional statements or triggers. Once these conditional declarations are encountered, actions or functions are carried out. Entire event sheets can be re-used by other levels, instead of re-creating events for other levels. CONSTRUCT2 has been a more efficient and succinct way of developing games than that using the HTML5 platform presented by (Alom et al. 2015). Behaviours work as embedded functions that have been assigned to objects and can be reused whenever needed. It includes movements such as eight directions, platform, advanced features like physics and path finding; and useful utilities like fade, flash, wrap, pin and drag-drop.

The WebSocket plugin was used for bidirectional instantaneous communication between servers and clients on CONSTRUCT2 platform. The WebSocket protocol supports WebSocket conditions (Is connecting, Is open, Is supported, On message) and WebSocket actions (close, connect, send text). Since WebSockets are standards-based, it is compatible with any standards-compliant WebSocket server; it employs a JavaScript interface to enable the communication between students allowing for resource sharing of graphical or textual information, promoting in-game collaborations.

There are over 70 WebGL-based (Web Graphics Library) pixel shader effects to change, distort, blend, blur, mask, re-colour and more. To develop these games, Web Graphics Library was added to objects, layers and layouts for quick special effects. CONSTRUCT2's primary export platforms are HTML5 based. It is also supported by multiple browsers and devices ensuring accessibility. All the original ACT21S™ games were exported from the HTML5 platform to the CONSTRUCT2 platform.

2. GAME DESIGN

2.1 Platform

The game works on a host authoritative model, reporting only events, using guaranteed order delivery messages, with the client being fairly limited to issuing the host requests, and waiting for instructions. When the game starts it does not know if it's the host or a peer. Rather than require the game logic to test throughout if it is host or peer, it has been designed so that the game contains two sets of common functions (one for host, one for peer) and the unwanted set is disabled once the game determines its role. This way a common interface is presented for the rest of the game, and no logic is required to determine if the game is host or peer, the same function calls should simply work regardless. A set of functions is used that map to the common database events.

The game waits for a player to give a user name. It then tries to connect to the server. If the game is the first to connect it becomes the host, and disables peer specific code. If the game is the second to connect it becomes the peer, and disables host specific code. The game waits in the lobby for second player. Chat is enabled when the second player arrives. GotoLayout function is then used to move between the game pages. WebSocket plugin has been installed on CONSTRUCT2 platform. It is useful for real-time web applications, as in this study, since it employs 'push technology' over Web Socket allowing both students and the server to communicate freely and at efficient speed.

The games were intended for international use and were translated into several languages. The language settings of the content are not dependent upon browser language detection, but upon the users' preferred language which can be pre-selected. If the preferred language is not specficied by the user, then the games are presented in English. The multiplayer architecture is hosted on a remote server and includes several components including Linux, Apache HTTP Server, MySQL and PHP (Alom et al. 2015). The database is presented as a relational structure and the application packages are configured to support the various target languages and output, the resultant game view at the client's end. The Server is set up just as before, a MySQL database, with PHP support, on a server that allows the PHP to be called externally. CONSTRUCT2 uses an ajax post call to send JSON encoded data to call a small common PHP file 'construct2dbhandler.php' which then interfaces with the database. The file needs to live in the same directory as 'config.php', as the details in this are used by the 'construct2dbhandler.php' to establish the database connection.

The games are used in educational settings, and most commonly in schools. On completion of one game the students are redirected to the game dashboard with the option to select another game.

2.2 Game Engine

The multiplayer game structure is presented in Figure 1. The collaborative environment includes some essential steps to give users access to the games. Students log into the system using unique logins, and each pair is connected by a unique team code. On successful login, the student arrives at a virtual room, referred to as the 'Game Lobby' and is presented in Figure 2. Once students select a game they are provided with options to select an arrow in the 'Game Lobby'. After selection of the arrow by any student, both of them are moved through to the 'Game Room' on the basis of the correct combination of unique game identifiers. The student who first clicks the arrow to enter the 'Game Room' is informed as student A, the other student is by default referred as student B.

Unlike many traditional development environments, CONSTRUCT2 avoids selecting specific instances of objects when adding events, in favour of filtering through all instances of an object type on screen. When adding events, the editor allows the user to specify conditions or checks that must be fulfilled by each object instance on the screen before the event will be added or run by it. Events can be chained together using sub-events, allowing for more complicated behaviours to be created.

2.3 Game Design

CONSTRUCT2 has been used to develop these games. The basic framework of our games consist of 4 event sheets and two layouts are presented below.

- Login Events
- Lobby Events
- Common Function Events
- Language Events
- Login Layout

- Lobby Layout
- Global Layer

Each game uses the basic framework and also specific event sheets; layouts have been used to complete the game. The login event sheet is based on the condition for a student to give a username. It then tries to connect to the server. The lobby event sheet is designed in such a way so that the game waits in the lobby for second student. Once the second student provides the username and joins in lobby, the system is moved into the game room.

The Common Functions event sheet uses two forms of communication, these services provide the names servers used to negotiate a new game, before establishing direct peer to peer communication. By default, it will try to use Web Real-Time Communication (WebRTC) and the inbuilt CONSTRUCT2 multiplayer functions, however if the game is being run on an IoS (originally iPhone OS) device the game will use photon and WebSockets as the communication layer. This is done by the common function system test "Is on platform iOS", upon start of layout. All the layout pages describe the design pattern of the game.

The WebSocket allows interaction between the students and the server, facilitating real-time communication. This connection enables information and messages to be sent back and forth between the server and the client browser without restricting information flow. This presents opportunities for each student to interact with the objects or resources presented to them. When an action is completed by the student, the game sends a message through the WebSocket to the server, and the server then sends that message to the other client, the other student, or partner to inform them of the action. A common functionality across the games is the dragging and dropping of objects or resources within and between student screens. The framework handles common functionality as described below:

- login
- game establishment
- game synchronisation
- chat
- game communication
- data recording
- localisation

Each game in the ATC21S™ project uses login function for game establishment. Once the game is established then the system attempts to use the function for game synchronization in particular the decision for the student to be host and peer. Each game presents an embedded 'chat box' to enable both students to communicate via text during game play. While paired under a session, students can collaborate or operate between themselves within the game. The chat box is common for all layouts of the game, so is set as a global layer and is tracked automatically. The template uses the Rex CSV plugin, to store the contents of a CSV localization file.

During game play every action (selection, movement and clicks) and communication completed by each student is captured in a log file in a time linear structure. It is important that every event is captured in the database as each action and communication, even if ineffective for collaborative problem solving, is used to interpret the students' performance and experience in the game. Each game is presented similarly, with the content and context varying. An instruction stem is presented first, followed by a problem. Each game takes between 30 to 45 minutes to complete. Many of the games present asymmetrical perspectives, providing different information and resources to each student, thereby increasing their need for collaboration. Sharing resources and information between each student's screen or through the chat box is critical in building the students' understanding of the problem and ultimately solving it. The games also vary in difficulty level; some require less collaboration but are cognitively more difficult, while others are cognitively easier but require efficient collaboration to solve (Care et al. 2015). Students are encouraged to complete a minimum of three games, to allow for sufficient data to be captured for assessment purposes. The games are generally presented in increasing complexity, with the easiest game presented first, and the most difficult last.

In Figure 3, a portion of the 'Animal transfer algorithm' from the game 'Animal farm' is presented to describe how events, actions, and WebSocket have provided the functionality behind such game mechanics. A screenshot of the Animal Farm game is presented in Figure 4. The objective of the game is to help the farmer to move the animals and food across the river, the students can only move one at a time but be careful about what he leaves behind. The pairs have to recognize that the animals cannot be left alone with an animal that can eat them. The animals can be transferred with an empty trailer. Students can see the movement of the happiness indicator whether sad or happy, after the transfer of each animal to his partner.

The first portion of the algorithm in Figure 3, function 'P7DisplayAnimaltoHost' displays the animal ID on host by calling the function 'UpdateAnimalIdonPeer'. Function P7HideAnimaltoPeer hides the same animal on peer (to his partner) by calling the 'Remote Function'. 'TruckP7' displays the Truck on the host side and the action 'Move to Move to' sends the object to the specific position. The function 'MoveHappinessToHost' is used which moves the happiness indicator to different position for the host. The function 'MoveHappinessToPeer' has been used to move the happiness indicator to different position in peer.

A set of functions has been used to map the common database events. Different actions are captured by using for example, RecRetry(A/B) eg: function Call "RecRetry"("A") calls the function 'RecRetry' by student A, to record the information of retrying of the game. RecPassResource("B",resource,destination) is used to record the information into database for the action of pass resources in the game by student B; where resource represents the objects are being passed to destination.

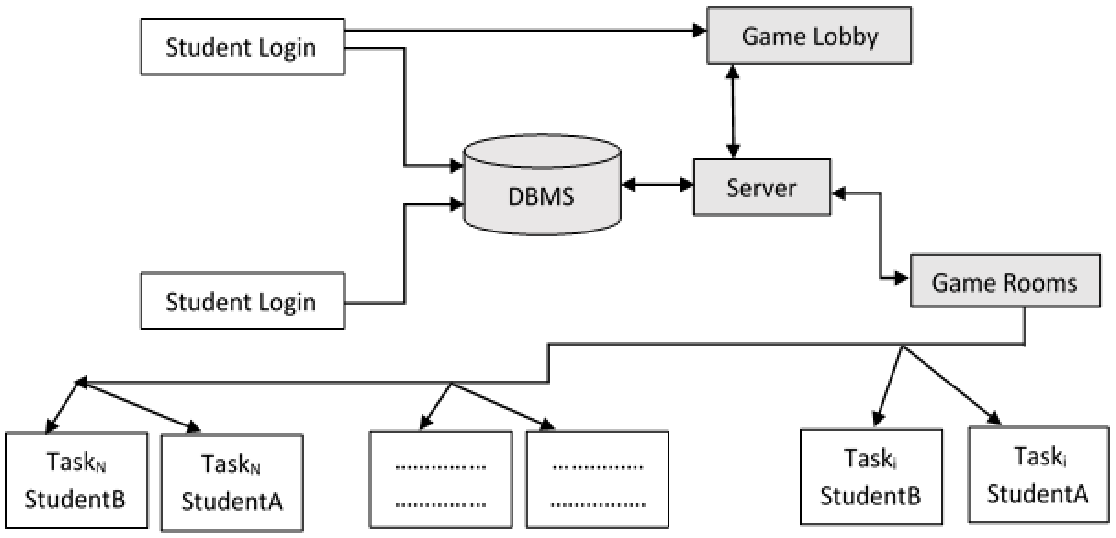


Figure 1: Multiplayer Game Structure (Adapted from Awwal *et al.* 2015)

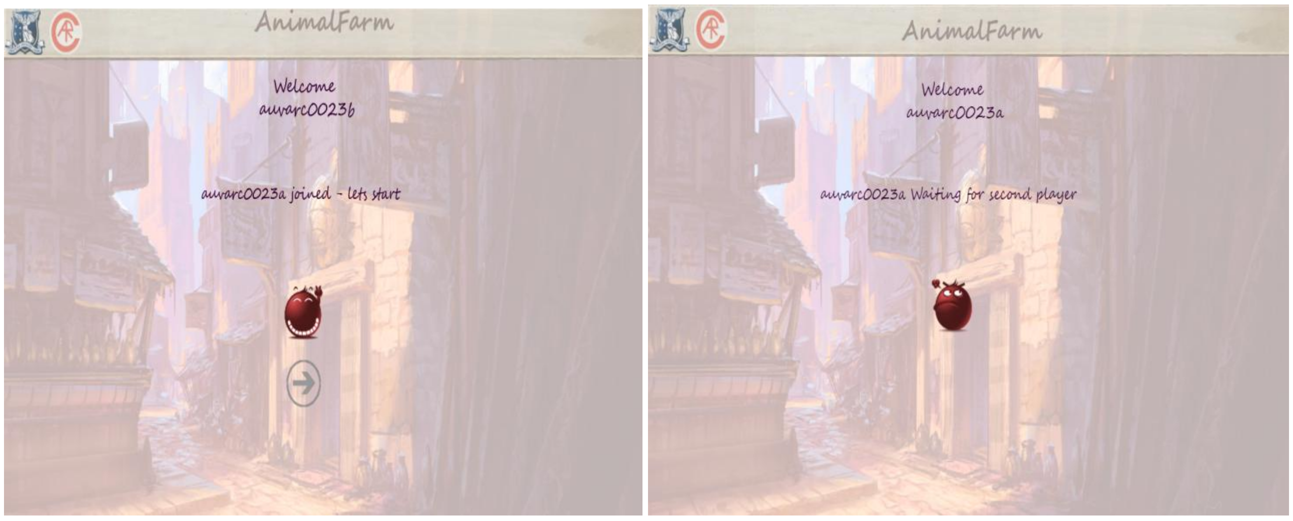
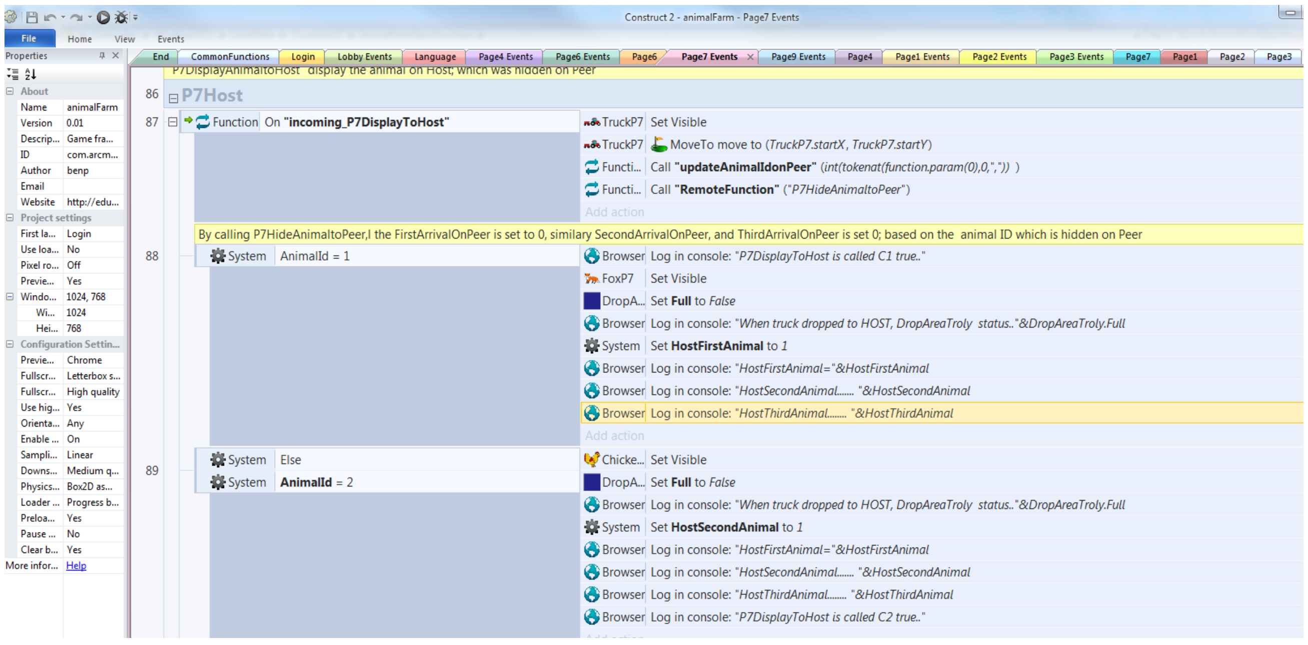

AnimalFarm
Welcome
auwarc0023b
auwarc0023a joined - lets start
AnimalFarm
Welcome
auwarc0023a
auwarc0023a Waiting for second player

Figure 2: Multiplayer Game Lobby Structure



Properties

About	
Name	animalFarm
Version	0.01
Descrip...	Game fra...
ID	com.arcm...
Author	benp
Email	
Website	http://edu...
Project settings	
First la...	Login
Use loa...	No
Pixel ro...	Off
Previe...	Yes
Windo...	1024, 768
Wi...	1024
Hei...	768
Configuration Settin...	
Previe...	Chrome
Fullscr...	Letterbox s...
Fullscr...	High quality
Use hig...	Yes
Orienta...	Any
Enable ...	On
Sampli...	Linear
Downs...	Medium q...
Physics...	Box2D as...
Loader ...	Progress b...
Preloa...	Yes
Pause ...	No
Clear b...	Yes
More infor...	Help

```
P7DisplayAnimaltoHost" display the animal on Host; which was hidden on Peer
86  P7Host
87    Function  On "incoming_P7DisplayToHost"      TruckP7   Set Visible
                                                   TruckP7   MoveTo move to (TruckP7.startX, TruckP7.startY)
                                                   Functi... Call "updateAnimalIdonPeer" (int(tokenat(function.param(0),0,",")) )
                                                   Functi... Call "RemoteFunction" ("P7HideAnimaltoPeer")
                                                   Add action
    By calling P7HideAnimaltoPeer,I the FirstArrivalOnPeer is set to 0, similary SecondArrivalOnPeer, and ThirdArrivalOnPeer is set 0; based on the  animal ID which is hidden on Peer
88      System  AnimalId = 1                       Browser   Log in console: "P7DisplayToHost is called C1 true.."
                                                   FoxP7     Set Visible
                                                   DropA...  Set Full to False
                                                   Browser   Log in console: "When truck dropped to HOST, DropAreaTroly  status.."&DropAreaTroly.Full
                                                   System    Set HostFirstAnimal to 1
                                                   Browser   Log in console: "HostFirstAnimal="&HostFirstAnimal
                                                   Browser   Log in console: "HostSecondAnimal....... "&HostSecondAnimal
                                                   Browser   Log in console: "HostThirdAnimal........ "&HostThirdAnimal
                                                   Add action
89      System  Else                               Chicke... Set Visible
        System  AnimalId = 2                       DropA...  Set Full to False
                                                   Browser   Log in console: "When truck dropped to HOST, DropAreaTroly  status.."&DropAreaTroly.Full
                                                   System    Set HostSecondAnimal to 1
                                                   Browser   Log in console: "HostFirstAnimal="&HostFirstAnimal
                                                   Browser   Log in console: "HostSecondAnimal....... "&HostSecondAnimal
                                                   Browser   Log in console: "HostThirdAnimal........ "&HostThirdAnimal
                                                   Browser   Log in console: "P7DisplayToHost is called C2 true.."
```

Figure 3: Portion of Animal Transfer Algorithm in 'Animal Farm' Game

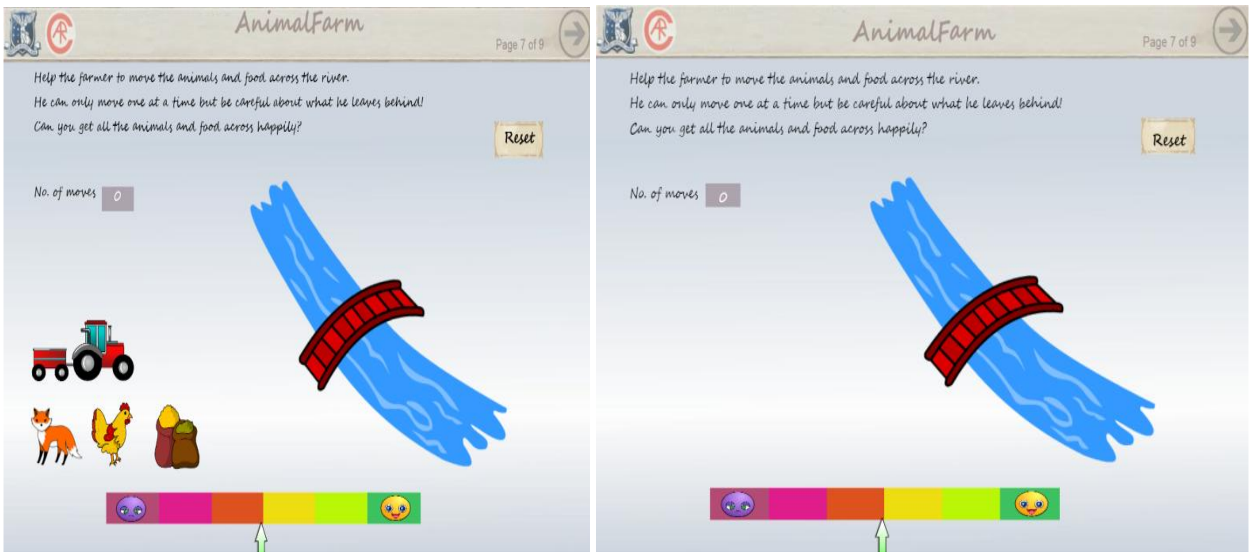

AnimalFarm
Page 7 of 9
Help the farmer to move the animals and food across the river.
He can only move one at a time but be careful about what he leaves behind!
Can you get all the animals and food across happily?
Reset
No. of moves
0
AnimalFarm
Page 7 of 9
Help the farmer to move the animals and food across the river.
He can only move one at a time but be careful about what he leaves behind!
Can you get all the animals and food across happily?
Reset
No. of moves
0

Figure 4: Student A's screen view on left and Student B's screen view on right of Animal Farm game

3. EXPERIMENT AND RESULTS

The necessity for development of games of this complexity is efficiency. Table 1 presents a comparison of development time and coding length of two if the ATC21S™ games: Olive Oil and Balance Beam (see Care et al. 2015 for full game descriptions). Olive Oil is a single page game initially built in HTML5 platform using Phaser library. Balance Beam is a five page, slightly more difficult game initially built in HTML5 using the KineticJS library. As can be viewed in the table, the time taken for the development of Olive Oil in the CONSTRUCT2 platform was three days, a significant reduction to the original development time of one month. Similarly, the Balance Beam development time was reduced from two months to two weeks when redeveloped on the CONSTRUCT2 platform. The coding length for each game was also reduced considerably. The shorter of the two games, Olive Oil, was reduced from 3738 lines to 1723 lines. The longer game, Balance Beam, has five pages and was reduced by more than a 1000 lines of coding by redeveloping it on the CONSTRUCT2 platform. The streamlining of platform indicates massive efficiencies. Time, and subsequently cost, can be reduced allowing for these allocations to be placed elsewhere in the study.

Table 1. Comparison of game completion time and coding complexity

Platform	**Game name**	**Time Taken**	**Coding Length**
HTML5	Olive Oil	One month	3738 lines
Construct2	Olive Oil	Three Days	1723 lines
HTML5	Balance Beam	Two months	29369 lines
Construct2	Balance Beam	Two weeks	17913 lines

4. CONCLUSION AND DISCUSSION

A key criterion of the ATC21S™ project is to develop interactive games that use synchronous communications to provide support for a collaborative process. CONSTRUCT2 is an HTML5 based 2D game editor is used to develop games; which allows quick creation of games in a drag-and-drop fashion using a visual editor and a behaviour-based logic system. Event logic such as OR and AND, as well as sub-events (representing scope) allow for sophisticated systems to be programmed without learning a relatively more difficult programming language. CONSTRUCT2 is quick and easy to bring works to life in hours and days instead of weeks and months. Both the length of programming coding and difficulties of game structure have been optimized by using CONSTRUCT2 platform. In this paper the design and implementation of a portion of the game 'Animal Farm' on the CONSTRUCT2 platform is also described. Not only the games are redesigned but also new games are developed using this novel structure. To design multiplayer games that demand collaboration over single player functionality is not easy. However, as novel technologies such as CONSTRUCT2 and WebSocket have shown such development is possible while ensuring consistent game flow and a positive user experience.

Other research work of ATC21S^{TM} project regarding the ability assessment of the students through these online collaborative games will be presented in future as research is progressing.

5. ACKNOWLEDGMENTS

Our thanks to the ATC21S™ project members who have made tremendous effort to carry out the research work.

6. REFERENCES

[1] Alom, B. M., Awwal, N. and Scoular, C. 2015. Technology Integration in Multiplayer Game Design, in European Conference on Games Based Learning Norway, 10-14.

[2] Amresh, A., Clarke, D. and Beckwith, D. 2014. GameScapes and SimApps: New Techniques for Integrating Rich Narratives With Game Mechanics, Proceedings of the European Conference on Games Based Learning, 1, 18-25.

[3] Care, E., Griffin, P., Scoular, C., Awwal, N. and Zoanetti, N. P. 2015. Collaborative Problem Solving Tasks' in Care, E. and Griffin, P., eds., Assessment and Teaching of 21st Century Skills Volume 2 - Methods & Approach, Dordrecht: Springer.

[4] Construct2 2016. Scirra, [online], available: https://www.scirra.com/construct2

[5] Crockett, L. 2014. HTML5 Canvas, User Illusions and Game Flow, Proceedings of the European Conference on Games Based Learning, 1, 68-76.

[6] Marchetti, E. and Valente, A. 2014. Design Games to Learn: A new Approach to Playful Learning Through Digital Games, Proceedings of the European Conference on Games Based Learning, 1, 356-363.

[7] Masuch, M. and Rueger, M. 2005. Challenges in collaborative game design developing learning environments for creating games, Third International Conference on Creating, Connecting & Collaborating through Computing (C5'05), 67.

[8] Zoanetti, N. P. 2010. Interactive computer based assessment tasks: How problem-solving process data can inform instruction, Australasian Journal of Educational Technology, 26(5), 585-606.