JAMES COOK UNIVERSITY

COLLEGE OF BUSINESS, LAW AND GOVERNANCE

CP3003 – Web Technologies

ASSIGNMENT 2 – WEB2.0 PROJECT

JAKE DIXON, DANE LENNON & JOSH WHATMOUGH
BACHELOUR OF INFORMATION TECHNOLOGY AND BACHELOR OF ENGINEERING

WEEK 13 [MAY 16]

Pong Revised

Chapter 1: Project Objectives

The aim of the project was to reinvent the well-known arcade video game Pong. The game is one of the earliest arcade games where each player is presented with a two-dimensional ping-pong-like arena. The main aim is to defeat an opponent in a simulated table-tennis game by earning a higher score. This project aimed to meet these requirements of the game whilst also incorporating additional features and graphics effectively placing a modern twist on the retro game.

1.1 Project Web 2.0 Features

1. Interactivity: Paddles controlled via JavaScript logic and HTML5 Canvas libraries
2. Interoperability with dual players plus spectators: 2 player game and any other clients connected are spectators
3. Using standardized technologies such as html5 and JavaScript: Allow for easy design and implementation and simplicity in the code
4. Modularity and portability: The site can run on many different systems and browsers and can be easily changed or have classes swapped out if need be

1.2 Features Implemented

The completed project was able to implement a working two-dimensional table-tennis arena with multiplayer ability via network connection. The following outlines the extra features that were achieved in the design phase of this project.

1) A two-dimensional table tennis arena
2) Event driven, user controlled paddles via keyboard and mouse input
3) Multiplayer support via network connection
    a) Player 1 hosts, player 2 joins
    b) Server is an intermediate between the two
    c) Server controls what data is sent where to cut down on lag
    d) Spectator mode for a 3rd client to connect to
4) Simulated ball physics implementing table-tennis-like rules

5) Game effects
    a) Performance Penalty: Speed of ball increases with game progress
    b) Comet Tail: Ball has comet tail showing path, the tail increases as the speed increases
    6) Scoring system to track game progress
    7) Heads Up Display to display messages to the players
8) Power ups
    a) Fast-ball: Speed of ball temporarily increases
    b) Multi-ball: ball divides by 3

Chapter 2: Features Planned but not Implemented

The project phases were successfully completed in accordance with the project plan. However, some features were not implemented due to the time constraint and complexity. These included:

1) Implementation of more power up classes
    a) Paddle length increase
    b) Opponent's paddle length decrease
    c) Weaponised paddle to attack opponent (opponent must avoid while also returning ball)
    d) Paddle Magnetized (ball attracted to paddle)
    e) Take Flight (paddle can be removed from back wall and move in a two-dimensional way)
2) Random events
    a) Earthquake - Screen shake
    b) Stuck-in-the-mud mode (sluggish movements of paddles)
    c) Mind-of-its-own (erratic movements of ball)
3) Round-the-world story mode
    a) Use of themes and borders to portray journey around the world as game progresses
4) Database to store user data.

Chapter 3: Potential Improvements

The project was able to implement a stable version of the Pong Revised game. During the testing phase there were some problems identified which included latency in the network connection and the lack of power-ups during game play. These are identified areas to improve and have been prioritised for future implementations. Version 2.0 of the Pong Revised will include improved latency over the socket.io connection reducing the observed lag for the multiplayer component of the game. This would be achieved by reducing the data packets sent and received across the network connection and implementing a smarter syncing strategy for the multiplayer game play. Future design would also ensure the addition of the features that were planned but not implemented. This includes the power-ups not developed and the around-the-world game play.

Chapter 4: Framework and Technologies Adopted

The design phase of the project included the use of a number of platforms and technologies creating a rich Web2.0 implementation. The Pong Revised game features the use of:

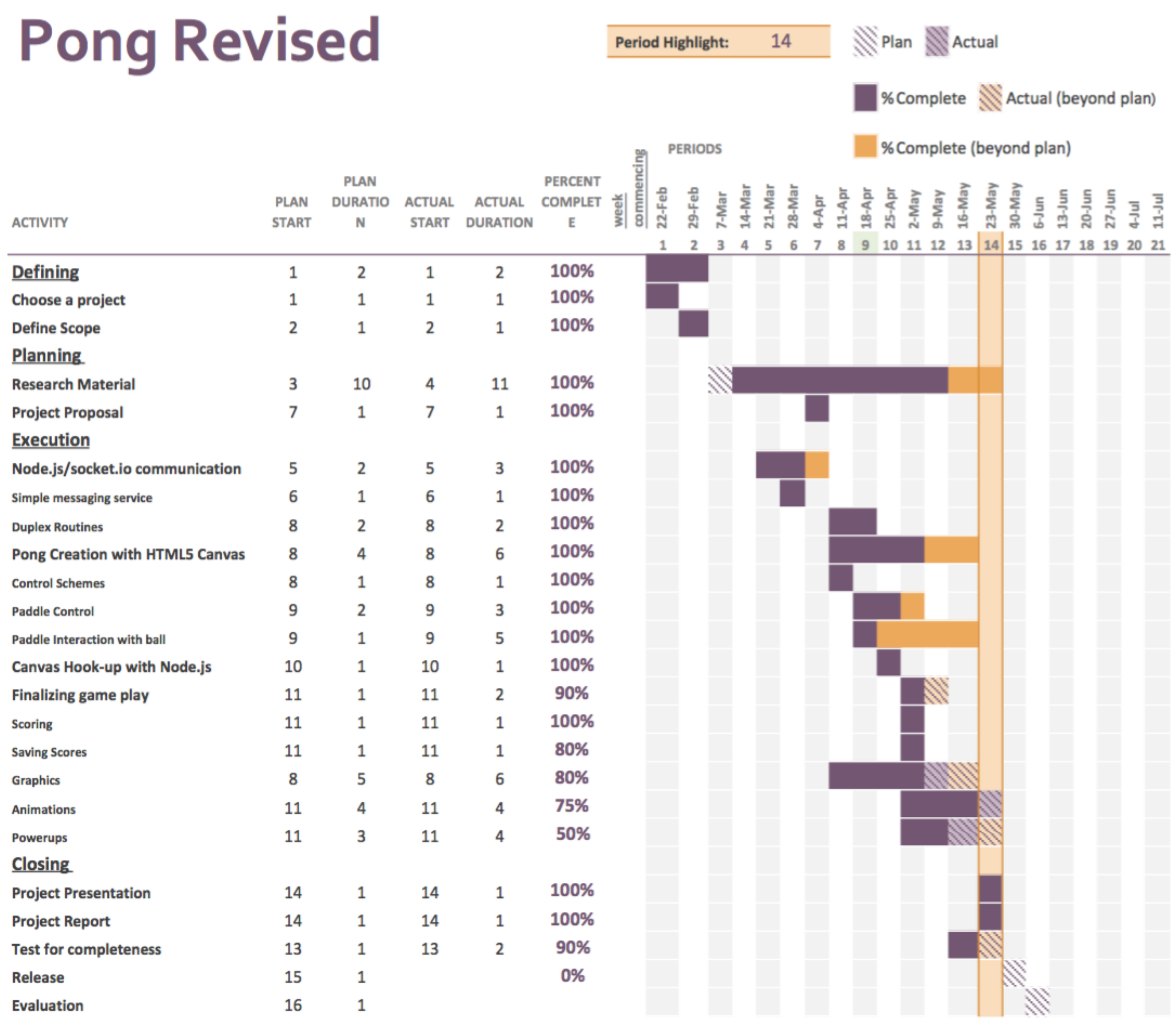
1) Node.js: Executes the socket.io server
2) Apache: to serve the html and JavaScript files, essentially to host the game as a site
3) Socket.io: used for multiplayer connectivity and synchronisation
    a) Server controls the data received and sent from each client
    b) Used to cut down data load between each client
    c) Examples are:
        - Player 1 doesn't need to receive paddle object from the server
        - The HUD doesn't need to be sent or received if it hasn't been updated
        - Controls the client's "player number" and refreshing of the pages
        - Controls connection and disconnection
4) HTML5 Canvas: use for the two-dimensional graphics and game renderings
5) JavaScript: for the majority of game logic and handling input and output
6) CSS: Basic text styling and custom fonts
7) JSON: easily transferrable objects that are passed between each client
8) Development Software / tools:
    a) Visual Studio Code and other code editors such as brackets
    b) Google Chrome and other web browsers
        i) Developer tools
    c) GitHub: https://github.com/joshuaWhatmough/alphaPongRevised
    d) Site: (server may go offline at any time): http://ditwebtsv.jcu.edu.au/~jc259368/pong
    e) Presentation: http://prezi.com/waoqliv2ldvh/?utm_campaign=share&utm_medium=copy

Chapter 5: Team Review on Individual Contributions of Each Member

Member	Leadership	Technical Contributions	Skill Contributions	Overall
Jake Dixon	⅓	½	⅓	40/100
Dane Lennon	⅓	¼	⅓	30/100
Joshua Whatmough	⅓	¼	⅓	30/100

Chapter 6: Actual Implementation Schedule

Initially, the project requirements outlined a design that involved condensing the code into a single file. However, throughout the design cycles, this method proved difficult to read. It also increased the complexity required to debug and refactor. The implementation was then shifted to an Object Oriented Programming (OOP) design structure. Each element within the game was designed to have its own class ensuring abstraction and proper development methods. OOP design made it easier to implement new features and swap out classes if a better one is written.

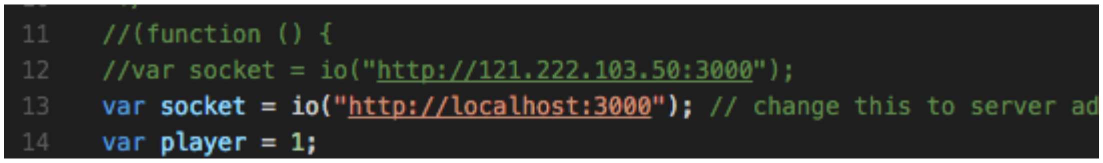


ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Defining	1	2	1	2	100%
Choose a project	1	1	1	1	100%
Define Scope	2	1	2	1	100%
Planning					
Research Material	3	10	4	11	100%
Project Proposal	7	1	7	1	100%
Execution					
Node.js/socket.io communication	5	2	5	3	100%
Simple messaging service	6	1	6	1	100%
Duplex Routines	8	2	8	2	100%
Pong Creation with HTML5 Canvas	8	4	8	6	100%
Control Schemes	8	1	8	1	100%
Paddle Control	9	2	9	3	100%
Paddle Interaction with ball	9	1	9	5	100%
Canvas Hook-up with Node.js	10	1	10	1	100%
Finalizing game play	11	1	11	2	90%
Scoring	11	1	11	1	100%
Saving Scores	11	1	11	1	80%
Graphics	8	5	8	6	80%
Animations	11	4	11	4	75%
Powerups	11	3	11	4	50%
Closing					
Project Presentation	14	1	14	1	100%
Project Report	14	1	14	1	100%
Test for completeness	13	1	13	2	90%
Release	15	1			0%
Evaluation	16	1			

FIGURE 1 - PROJECT TIMELINE

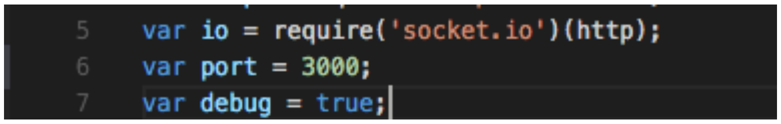
Chapter 7: Project Setup Instructions

1) The game uses a server to host HTML to each client
2) Server setup:
    a) Server needs to have web hosting capabilities
    b) Server needs to have node.js installed
    c) The server was configured to host the game html and JavaScript files to each client
    d) The server also executed the node.js server.js file which is routed through a port to the clients
    e) The server awaits incoming connections
3) Client setup:
    a) So long as the server is configured correctly, a client should be able to connect to the server via it's IP or domain name and commence playing the game with another person
    b) Server configuration for localhost
        i) In the game.js file configure the line socket variable so that it points to the server's IP address



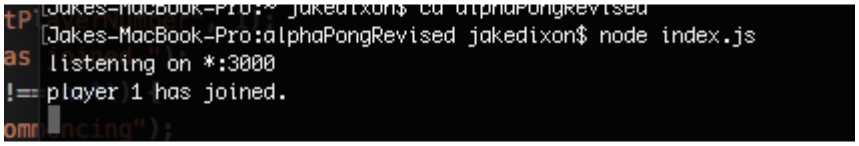
```
//(function () {
//var socket = io("http://121.222.103.50:3000");
var socket = io("http://localhost:3000"); // change this to server ad
var player = 1;
```

        ii) In this case the server is the localhost and the port is 3000
        iii) In the index.js file set the port to 3000



```
var io = require('socket.io')(http);
var port = 3000;
var debug = true;
```

        iv) Execute the NodeJS server with the command "node index.js"
        v) Wait till the output shows that the server is listening to the port
        vi) Open the index.html file and the server output should show that you have connected.



```
Jakes-MacBook-Pro:~ jakedixon$ cd alphaPongRevised
Jakes-MacBook-Pro:alphaPongRevised jakedixon$ node index.js
listening on *:3000
player 1 has joined.
```

4) Server configuration for outside server (may not work the same for everyone)

a) Make sure that the game files are in a directory that is hosted from the server, for example the public/www directory

b) Edit the game.js file so that the socket variable points to the external IP of the server

c) Forward the port 3000 through your router

d) Execute the node.js server

e) Type the server's external IP into a browser and it should load

Appendix A - Reflective Journal

Date	**Git Commit Message**	**By**	**Remarks**
7 Apr 16	Initial Commit	Joshua Whatmough	Project commenced
8 Apr 16	Add functionality to ensure paddle does not draw off screen	Dane Lennon	Observed paddles sliding out of bounds
10 Apr 16	Add mouse paddle control	Dane Lennon	Mouse events used. Keyboard control was superseded
10 Apr 16	Add socket.io dependencies and the chat prototype files for references	Jake Dixon	Basic server app and connectivity between two clients. Chat prototype for reference
10 Apr 16	Integrate CSS into html Add socket.io server script index.js Add socket.io client script	Jake Dixon	Implemented logic to parse the html.
12 Apr 16	Latency Test	Jake Dixon	Testing the latency
12 Apr 16	Add maths for determining where on the paddle the	Dane Lennon	Paddle divided into 3 sections

	ball hit		
12 Apr 16	Hit detection on 2nd paddle	Joshua Whatmough	Player 2 paddle hit detection logic implemented
13 Apr 16	Add basic 2p play	Jake Dixon	Passing paddle data between two clients so that each can move a paddle
13 Apr 16	Add Paddle.js	Dane Lennon	Commenced separation of class files
26 Apr 16	Ball class completed	Jake Dixon	Separated game into classes for OOP
27 Apr 16	Add documentation to the ball class	Dane Lennon	Documented using JSDoc
27 Apr 16	Ball bouncing off paddles and hit detections	Jake Dixon	Modifications to the way the ball is deflected
27 Apr 16	Amend hitTest Parameters to ensure paddle does not rely on knowing about ball class	Dane Lennon	Instead of ball object passed to paddle hitTest, the ball x and y coordinates were passed to paddle
27 Apr 16	making some OOP	Jake Dixon	Adding some more separate classes
27 Apr 16	multi ball!	Dane Lennon	Multi-ball functionality commenced
27 Apr 16	Adjust paddle height limitations	Joshua Whatmough	Ensure paddle can reach sides of game arena
27 Apr 16	Made the number of balls	Jake Dixon	Added some parameters to

	modular		support power ups
27 Apr 16	Major additions for server side: 2p paddles, server side backend for ball and paddle control	Jake Dixon	Halved the number of requests to the server for the ball passing and reduced the paddle data by 90%
27 Apr 16	Start p2 hitTest	Jake Dixon	Re-implementing the 2nd paddles hit testing
28 Apr 16	2nd paddle hitTest now works	Jake Dixon	Fixed a few bugs belonging to the hit detection
28 Apr 16	Add player detection for connecting / disconnecting	Jake Dixon	add a wait for 2nd player to join, add a disconnection listener so the game stops if p1 or 2 disconnect, add hit detection on p2 paddle (needs working on), add spectator mode
29 Apr 16	Add scoreboard / HUD	Jake Dixon	Basic implementation for keeping score and displaying messages
03 May 16	small aesthetic updates and documentation	Jake Dixon	Added a custom font and changed some colours.
03 May 16	Add ball comet tail	Jake Dixon	Implemented a tail that follows the comet. Basically just an array of the previous 10 positions
04 May 16	add calcInitialVelocity function and add testBall.html file	Dane Lennon	Fixed the initial position and direction of the balls

04 May 16	Fixed hitTest2 function	Joshua Whatmough	Tweaking the p2 paddle hit test again
04 May 16	add getHitPosition function for paddle 1	Dane Lennon	getHitPosition function created for reuse
04 May 16	Started a PowerUp Class and multiball extension,	Joshua Whatmough	Also added a factory class for powerUp generation
04 May 16	Move powerup classes to their own folder	Joshua Whatmough	This was done to help reduce clutter
04 May 16	Halved the data going to server	Jake Dixon	Once again improved the server code so now each player only received the data it needs
04 May 16	server side optimizations for paddles	Jake Dixon	P1 doesn’t need to know about p1’s paddle from server
11 May 16	added muitiball support code	Jake Dixon	Observation made that more than 10 balls across server will cause lag
11 May 16	rewrote server connection	Jake Dixon	Complete rewrite of the way each client connects to the server. Stores session ids and can handle page refreshes now
18 May 16	power up class move now works	Joshua Whatmough	Power up now bounces from top to bottom
18 May 16	fastball.js added	Joshua Whatmough	Initial file skeleton for the power up

23 May 16	fix paddle bounce and resolve hitTest functions into single function	Dane Lennon	hitTest for both functions resolved into single function
22 May 16	started work on implementing powers	Jake Dixon	Got the power up to draw on the screen
24 May 16	Hittest for multiball, and multiball functionality working.	Joshua Whatmough	Ball can now hit the power up
24 May 16	MultiBalls now spawns on power up, rather than in the centre	Joshua Whatmough	Fixed the positions that the balls spawn when hitting a multiball
24 May 16	Fix ball respawn	Jake Dixon	Ball was going off screen and not respawning
24 May 16	fix scoring of MB and paddle bounce speeds in y direction	Dane Lennon	Ball now bounces faster in y direction for upper and lower parts of paddle
25 May 16	Fix MB spawning	Jake Dixon	Multiball now passed over the server so both clients support it
26 May 16	Add Fastball	Joshua Whatmough	The effect of making the ball fast is now implemented