REACTIVE SPACES API

DESIGN DOCUMENT | DESIGN STUDIO 4 | NOV 7 2014

FOR PROF. ALI ARYA | MARCO | MATTHEW | RYAN | ZARA

REACTIVE
SPACES

TABLE OF CONTENTS

1 INTRODUCTION

1.1 HIGH CONCEPT

Reactive Spaces is an API which allows for easy development of distributed javascript web apps that utilize the Microsoft Kinect.

1.2 API OVERVIEW

The API will consist of a desktop application, a server, and a JavaScript (JS) Library. The desktop app will manage a socket connection to the server and expose a local websocket connection over which to send and receive data with the JS library. It will be responsible for passing messages to and from the server and JS library as well as reading from and passing Kinect input data. The JS library will deal with connecting to and sending & receiving data from the desktop app. It will also define data types specific to the API as well as providing utility functions related to those data types. The server will then manage connections between all running instances of the desktop app, and match them into sessions based on app types.The server will also be responsible for passing messages between appropriate clients in each session.

1.3 GAME OVERVIEW

The game will aim to demonstrate all the features of the created API. The game will involve multiple stations participating and playing against each other. The game will be an arcade-style game where players aim to reach a high score. The game will use the API to provide input via the Microsoft Kinect. It will also use the API to share data between peers. The game will be influenced by the other stations who are also running the game.

2 API

2.1 GOALS

- Expose local kinect interaction data through a JS library
- Expose remote kinect interaction data through a JS library
- Allow for custom data to be sent between game instances
- Match and organize connections between game instances

2.2 SPECIFICATIONS

2.2.1 DESKTOP APP

PLATFORM / REQUIREMENTS

- Windows Vista or greater with .NET 4.5
- Kinect 360 sensor and Kinect runtime or SDK v1.8
- Internet connection

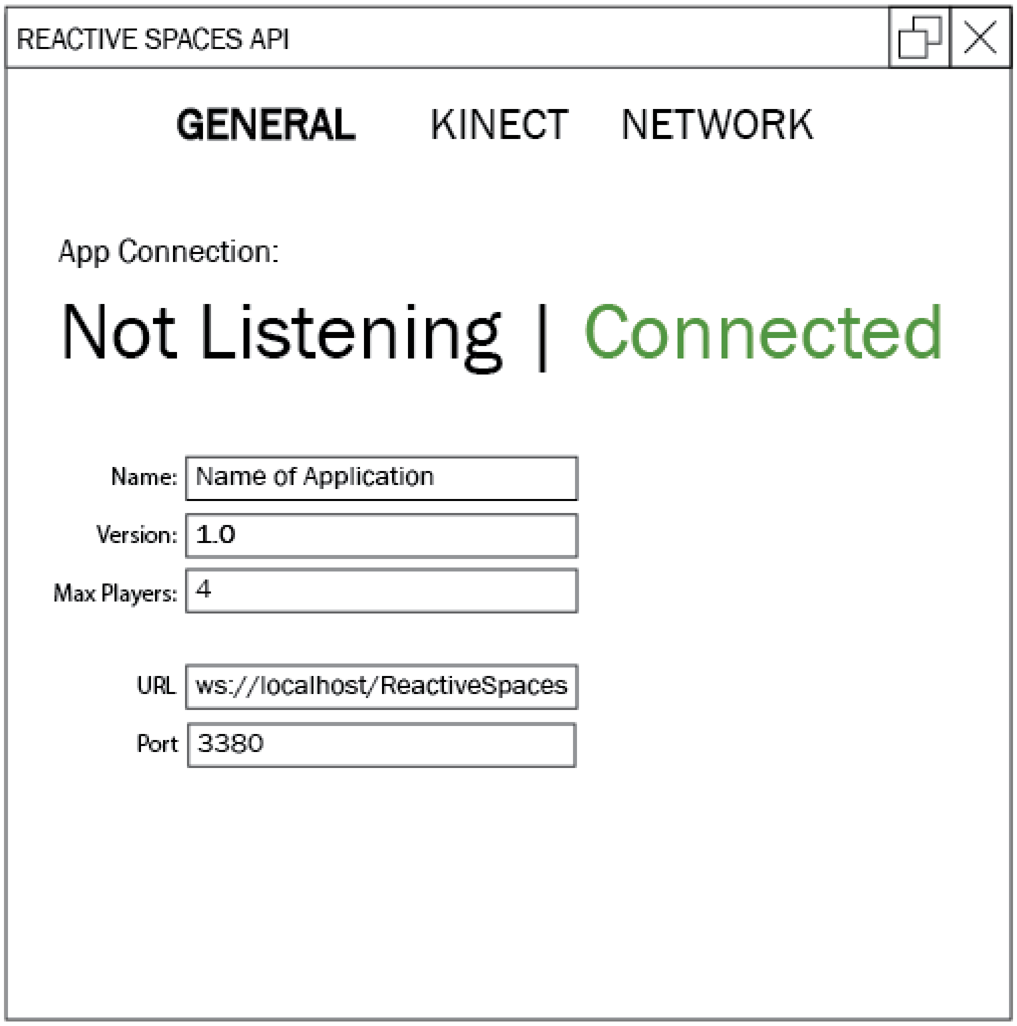
EXPOSED FUNCTIONALITY

- Set name and location information for the station
- See Kinect data and sensor status
- View connected application information and connection status
- View server connection status and information about stations in the same app session

INTERFACE

LAYOUT : GENERAL

This area contains information about the overall state of the app as well as information about the connected game instance and local websocket.



App Connection:

Not Listening | Connected

Name: Name of Application

Version: 1.0

Max Players: 4

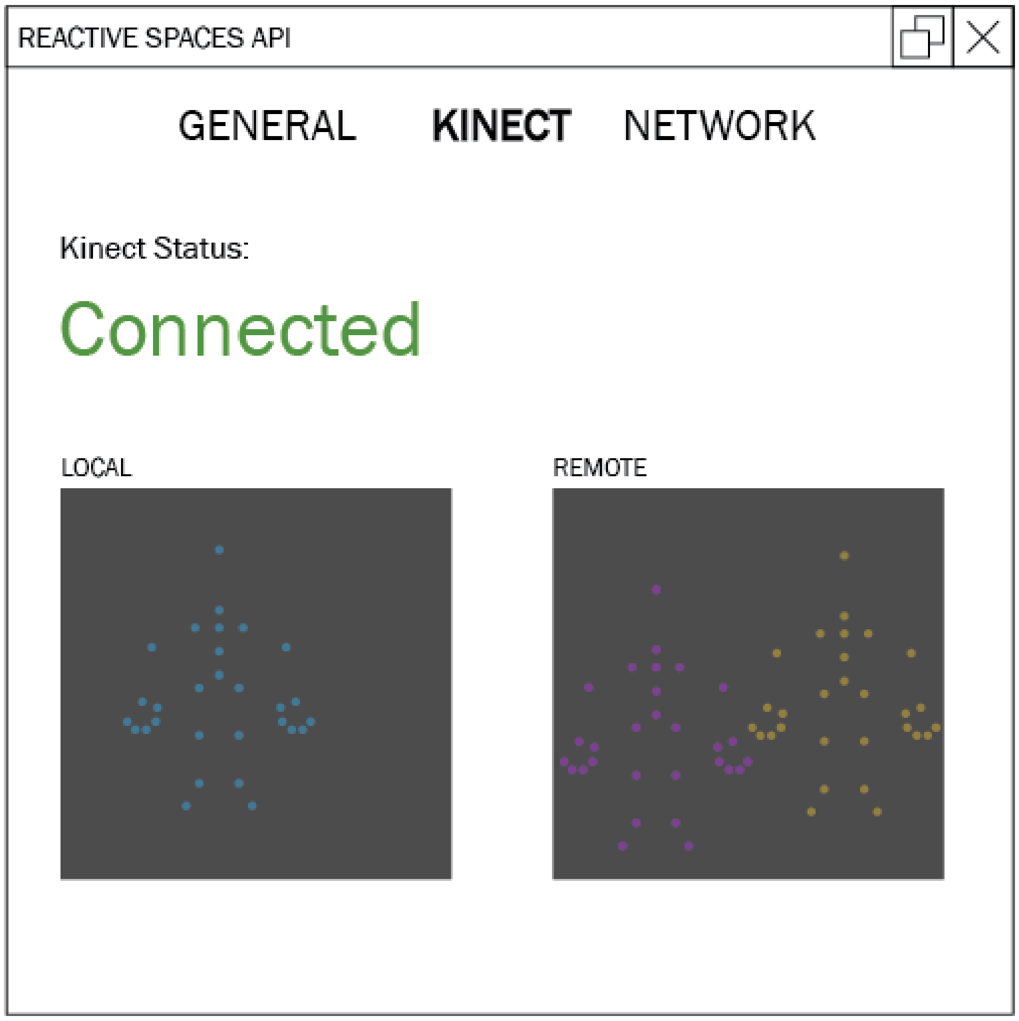
URL ws://localhost/ReactiveSpaces

Port 3380

REACTIVE
SPACES

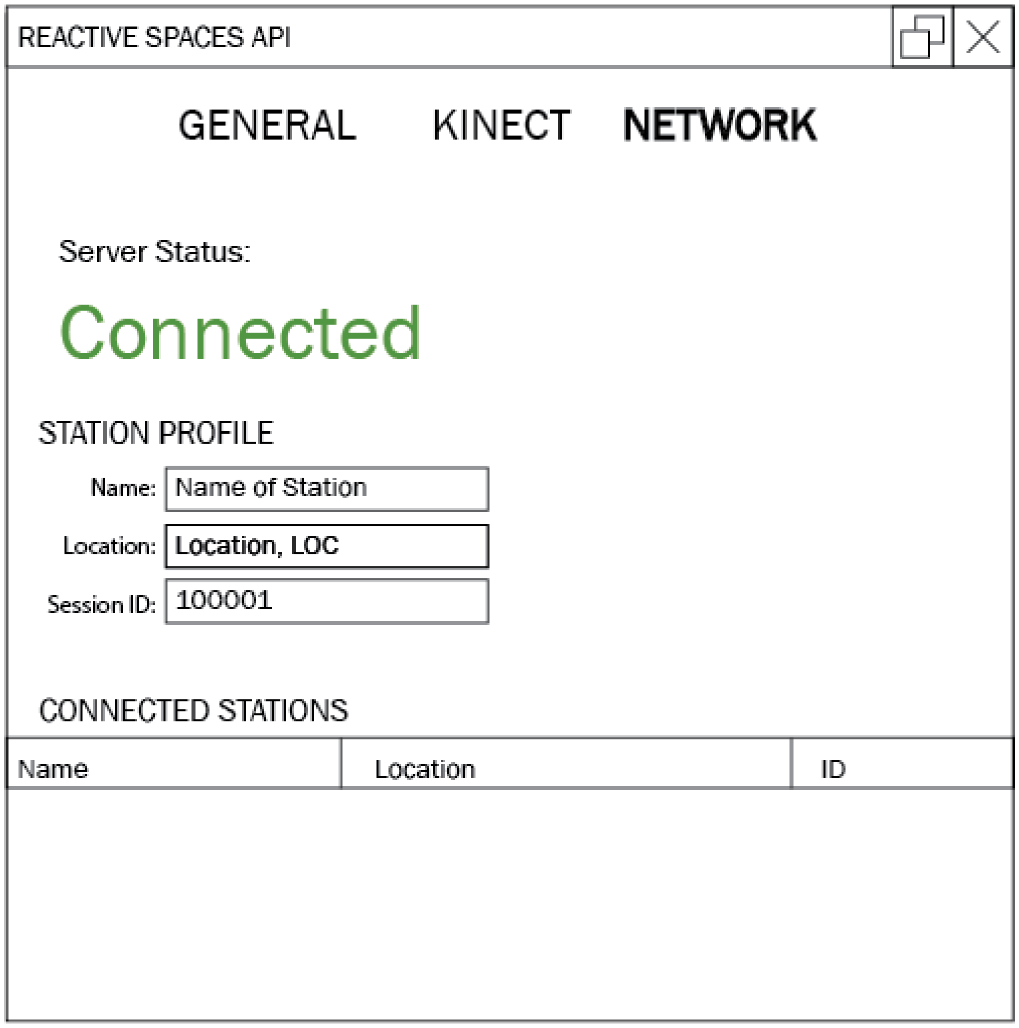
LAYOUT : KINECT

This area will display skeleton information for local and remote Kinect as well as an overall status of the Kinect system.



LAYOUT : NETWORK

This area contains information relating to the server connection. This is where users will enter information about their station. It will also provide and overall connection status as well as a list of connected peers.



REACTIVE SPACES API

GENERAL KINECT **NETWORK**

Server Status:

Connected

STATION PROFILE

Name:	Name of Station
Location:	Location, LOC
Session ID:	100001

CONNECTED STATIONS

Name	Location	ID

VISUAL STYLE : API

The desktop application will have a clean and memorable design. The colours shown in colour scheme below will be the main colours used throughout the application as well as the website, promotional materials, and the example game. The colours will consist of mainly shades of grey with the accent colours being shades of cyan and magenta.

#E7E8E8	#979898	#3D4142	#60C4B9	#B76BAB

VISUAL STYLE : LOGO

Below is the logo that will be used in the application. It will be used to create a brand out of the API.

MINIMAL :

MONOCHROME :

REACTIVE
SPACES

COLOURED :

REACTIVE
SPACES

2.2.2 SERVER PLATFORM

- JavaScript code running on Node.js server
  - Linux server
  - Node.js with Forever
  - Publicly accessible internet connection

FUNCTIONALITY

- Accept and manage connections from client apps
- Match clients into sessions based on connected games/apps
- Pass messages between clients in the same session
- Notify clients of changes in their session

2.2.3 CODE LIBRARY PLATFORM

- Web browser with web socket support
  - Internet Explorer 10 or greater
  - Firefox 31 or greater
  - Chrome 31 or greater
  - Safari 7 or greater
  - Opera 24 or greater

EXPOSED FUNCTIONALITY

A. MAIN FUNCTIONS

- Connect
  - Opens a connection to the desktop app, and sends information about the running web app.
- Disconnect
  - Closes the open connection to the desktop app.
- ActivateMessenger
  - If included in the project, this function creates an instance of the messenger object, which allows the JS library to display errors and messages on the webpage.
- addEventListener
  - Adds a listener to the JS library. Available event types are listed below in c. DataTypes.
- removeEventListener
  - Removes the given callback function from the active event listeners.
- Send
  - Sends a custom data object through the server to all other connected apps.

B. UTILITY FUNCTIONS

- DrawSkeleton
  - Draws a Skeleton object to the given canvas. This function is useful to get things off the ground quickly.

C. DATA TYPES

- Skeleton
    - Defines the data structure and provides functionality for Kinect skeletons passed from the desktop.
- SkeletonJoint
    - Defines the data structure and provides functionality for Kinect skeleton joints.
- Vector3
    - Defines the data structure and provides functionality for 3D vectors.
- StationProfile
    - Defines the data structure for station profiles.
- Features
    - Enumerator of possible app features that are required and / or supported. For now the only on e in Kinect.
- JointTypes
    - Enumerator defining the joint types that are available on the Skeleton object (ie elbow, shoulder etc).
- Events
    - Enumerating defining the events that the JS library fires. These are the events that user apps can listen to through the JS library.

3 DEMO GAME

3.1 GOALS

- Showcase the API that is created.
- Illustrate API features and functionality that will help game developers
- Show off how to use the API

3.2 MECHANICS

The game has two elements to it: The first being the primary form circles that the player can interact with. The player's goal is to collect a certain amount of the form circles which then turns into a larger circle. This larger circle collects the score triangle items to increase players score. The stations can also affect each others environment, making it harder for the other stations to play. When one station successfully creates a larger circle, it appears on all active stations. This larger circle helps the creator collect score not only on their station, but on all other stations. This circle can only be destroyed by crossing paths with a rival large circle created by another station. When this happens they will cancel each other out and both disappear.

3.3 GAMEPLAY

The players' main goal is to collect the highest amount of score possible. This is done by first collecting primary form circles to form a larger circle. This large circle then collects the score triangles which adds points to the station's score. This is done using the Kinect sensor. The players use their hands to control on screen representations of the hands to collect the primary form circles.

3.4 INTERFACE

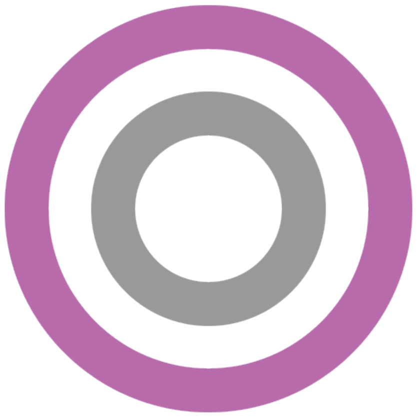
The game will have a simple interface for players. The interface will display the players score along the top of the screen. The menu for the game can be accessed by the player holding their hand in the top right corner. This will display all connected players scores.

3.5 ART STYLE

The game is going to represent the API's functionality so the art style is going to have the same branding elements as the API. The game will follow the colour scheme of the API using mainly different shades of grey with the accent colours different for each station. The style will also reflect the style of the gameplay is quite simple and abstract. The overall look of the game is a simplified outer space theme. The background will be solid black to allow the game elements to pop.

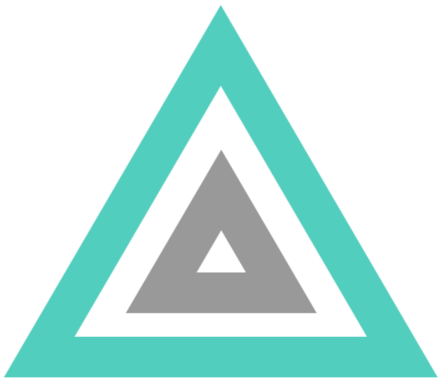
FORM CIRCLES :

The form circles will consist of two rings, one inner white ring and an outer coloured ring.



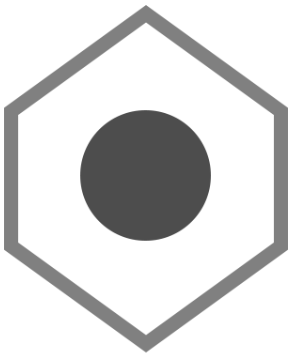
SCORE TRIANGLES :

The score items contrast the form circles in shape and colour. They will be triangular and a contrasting colour of the form circles.



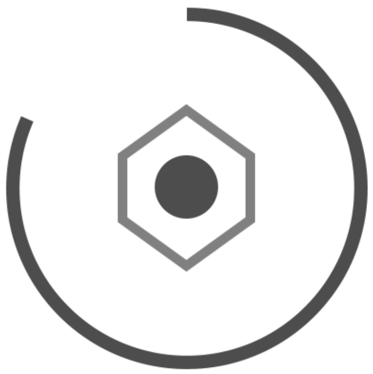
HANDS :

The hand symbols consist of a sphere and a hexagon around it. There will be two hand symbols per player.



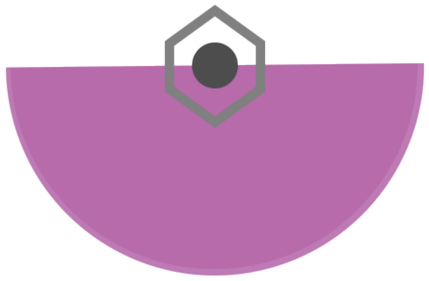
LOADING HANDS :

When loading for the menu to pop up, the loading circle will appear around the hand symbol.



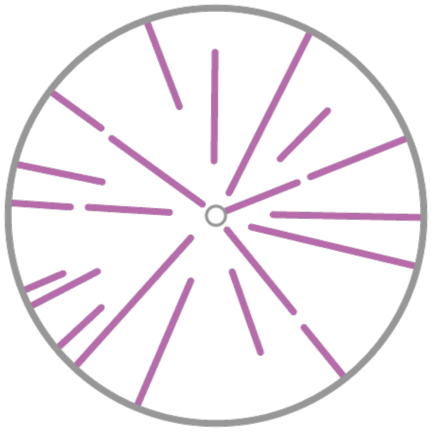
FILLING HANDS :

As the hand collects the form circles, the hand symbol will fill to create the large circles.



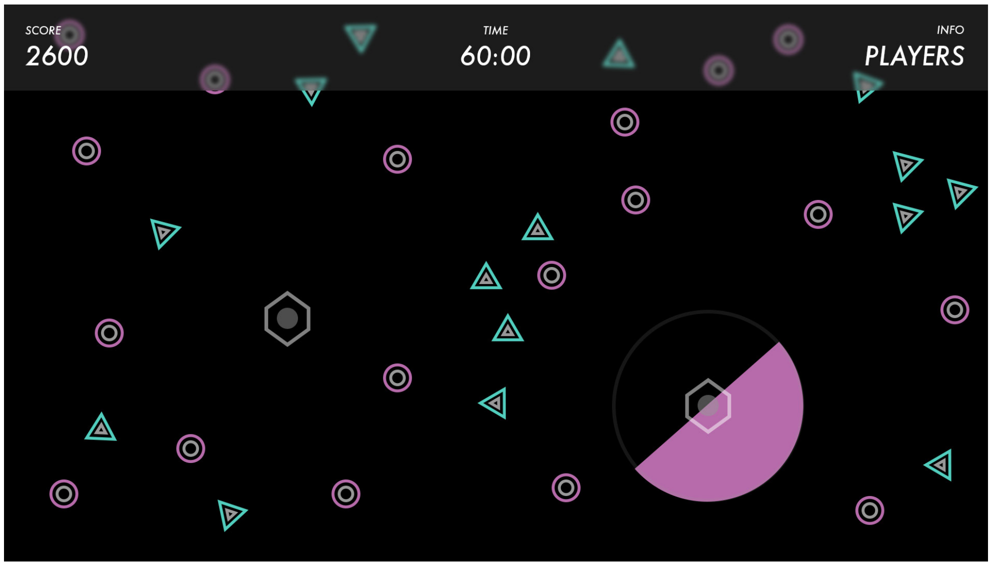
LARGE CIRCLES :

When the filling hands become full, they create a new large circle. Large circles also appear if they are created in another station's game, and these are differentiated by the color of the circle.

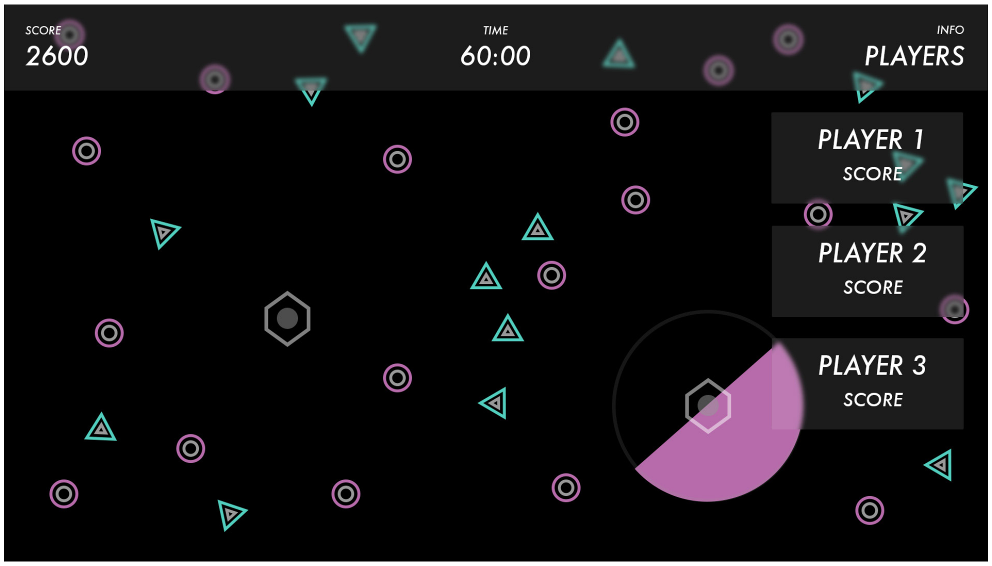


3.6 CONCEPTS

GAME :



MENU :

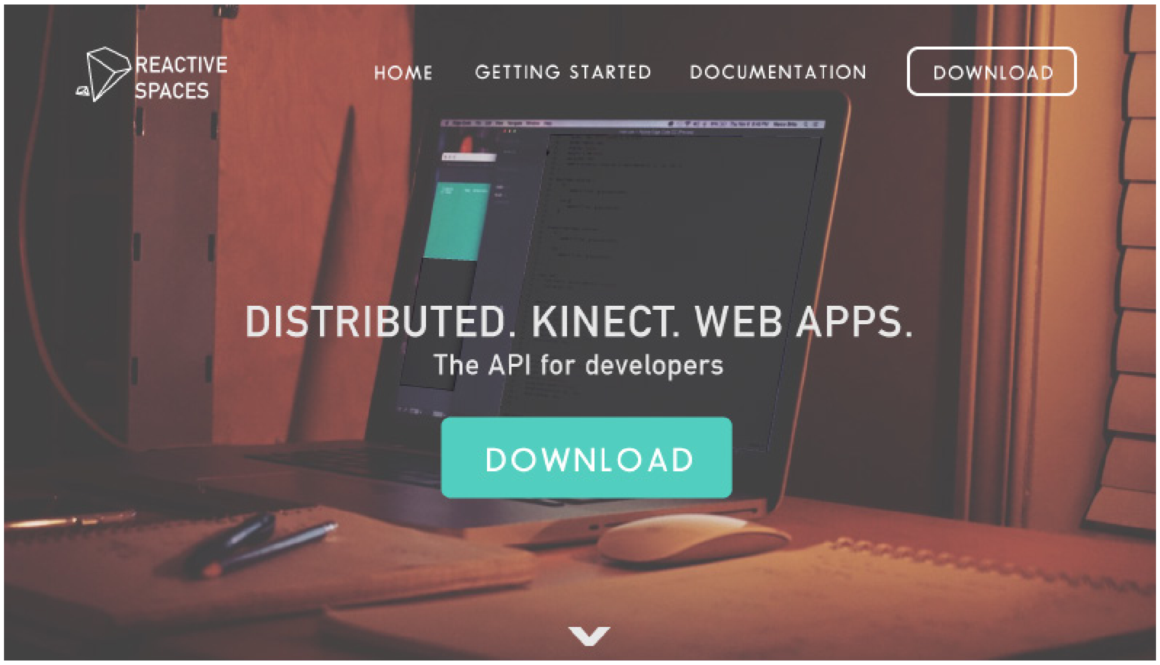


4 PROMOTIONAL PLAN

4.1 WEBSITE

The website will contain information about the API and example games developed with the API. There will be a download link for the API available on the website for developers to start developing with. The website will list the key features of the API, the documentation for developers, and contact information. The website has a colour scheme that will be incorporated in each of the promotion materials and match the design of the desktop API and the example game. The website will be the main medium for promotion.

ReactiveSpacesAPI.com (under development)



4.2 POSTERS / STICKERS

The posters and stickers will be used to create interest for programmers and to get them to visit the Reactive Spaces API website. From the team's experience, programmers love stickers. On the demonstration day the team will be handing out stickers for interested parties to keep. The posters will be used both electronically and physically to promote the API and live demonstration day.

4.3 LIVE DEMONSTRATION / DEMONSTRATION VIDEO

The demonstration day will be held on Monday December 1st and will be used to show off the API with the use of the example game. The demonstration will be stationed at the fourth floor university center from nine in the morning to three in the afternoon. There will be two televisions each running the game with kinect to show off the features of the API. During the live demonstration the team will be giving out info cards and stickers to whomever would like them. The demonstration will also be filmed in this time period to promote the API on the website.

5 LIST OF ABBREVIATIONS

- JS - Javascript
- p2p - peer to peer