DroidXTK

Jared Barresi
Jared.Barresi001@umb.edu
University of Massachusetts Boston

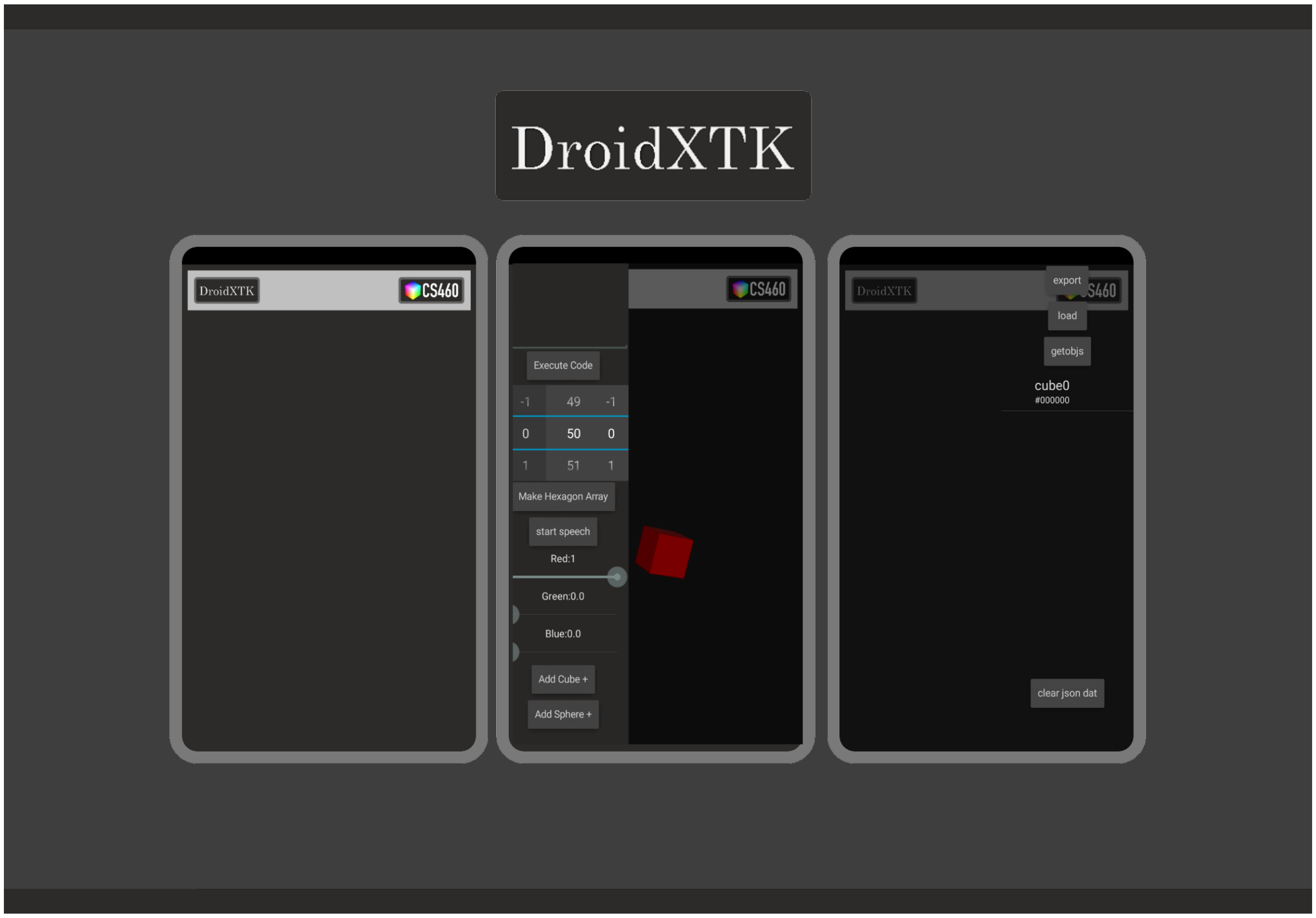

Figure 1: Create objects, use voice-recognition, and even debug code with the built in web console.

ABSTRACT

DroidXTK is an Android application and framework, of which provides a bridge (in JavaScript) to all the Android API's, for use with the XTK framework. The Android app is built using the JavaScript platform Droidscript. Some API "features" implemented include: Button/Slider input, Voice Recognition, Sensor i/o, and many more potential possibilities. The main concept will be to create generalized methods, of which can be used (as is) or easily be extended to provide new functionality in the future.

KEYWORDS

WebGL, Visualization, Android, JavaScript, XTK, Framework, Droidscript, Developer,API's, Voice-Recognition, Accessibility

ACM Reference Format:
Jared Barresi. 2019. DroidXTK. In *CS460: Computer Graphics at UMass Boston, Fall 2019.* Boston, MA, USA, 3 pages. https://CS460.org

1 INTRODUCTION

Using a webview container, you can modify the contents of a page easily via a console. Using an Android framework called Droidscript, you can execute custom-made codes in a webview that has the XTK framework loaded into it.

Scripting Engine:
The DroidScript App contains a scripting engine which allows anyone with a bit of JavaScript knowledge to easily write Apps for their mobile phone or tablet. You can write very simple Apps with just a few buttons, or more complex ones which include dynamic graphical interfaces such as the DroidScript application itself, which is written using the very same engine.

As well as creating graphical interfaces, you have access to Sensors like the Accelerometer, Compass, Light meter or other device components like Wifi, Bluetooth, Camera, GPS, SD Card, SMS, Emails, Internet and more. We're always adding new functionality to the engine, so if you want something added just let us know via email or leave a comment on the forum.

Source: www.androidscript.org

2 METHOD

App Object:

The **app** object is the main driving force behind Droidscript and its easy-to-use JavaScript app development framework. Some of the categories of what the app object extends to include:

- Application Control
- Application Information
- Bluetooth
- Components
- Controls
- Cross-Application
- Database
- Debugging
- Device Control
- Device Information
- Dialogs
- Files
- Graphics
- Layouts
- Messaging
- Network
- Sounds
- UI Control
- User Information

source : www.androidscript.org

2.1 Implementation

A "Nested Bridge"

$$\boxed{\boxed{Droidscript} \longrightarrow \boxed{Java} boxedJava \longrightarrow \boxed{AndroidAPI's}}$$

Executes JavaScript code on live HTML/JavaScript, in an Android webview container. XTK code bridge created in JavaScript to using Droidscript framework to develop methods.

Android Controls:

```
//controls to initialize methods
bexec = app.CreateButton("load");
btnClearJsonData = app.CreateButton("clear json dat");
btnCube = app.CreateButton("Add Cube +");
btnDebug = app.CreateButton("Execute Code");
btnobjlst = app.CreateButton("getobjs");
btnspch = app.CreateButton("start speech");
btnSphere = app.CreateButton("Add Sphere +");
btnwpcg = app.CreateButton("export");
createHexagon = app.CreateButton("Make Hexagon Array");
//data entry fields to store values to apply to XTK objects
edt = app.CreateTextEdit("",-1 ,0.18,"mono");
labelBlue = app.CreateText("Blue:0.0");
labelGreen = app.CreateText("Green:0.0");
labelRed = app.CreateText("Red:0.0");
//layouts to hold the prior objects
layDebug = app.CreateLayout("linear","VTop");
layDebug.AddChild(btnCube);
layDebug.AddChild(btnDebug);
layDebug.AddChild(btnspch);
layDebug.AddChild(btnSphere);
layDebug.AddChild(createHexagon);
layDebug.AddChild(edt);
layDebug.AddChild(layRGB);
layDebug.AddChild(layXYZPos);
layObjDebug = app.CreateLayout("linear","VTop");
layObjDebug.AddChild(bexec);
layObjDebug.AddChild(btnClearJsonData);
layObjDebug.AddChild(btnobjlst);
layObjDebug.AddChild(btnwpcg);
layObjDebug.AddChild(objlist);
layRGB = app.CreateLayout("linear","Vertical");
layRGB.AddChild(labelBlue);
layRGB.AddChild(labelGreen);
layRGB.AddChild(labelRed);
layRGB.AddChild(skbBlue);
layRGB.AddChild(skbGreen);
layRGB.AddChild(skbRed);
layXYZPos = app.CreateLayout("linear","Horizontal");
layXYZPos.AddChild(txtXPos);
layXYZPos.AddChild(txtYPos);
layXYZPos.AddChild(txtZPos);
//list to store and track objects created
objlist = app.CreateList(",",0.45,0.63);
//More controls to change properties post-running
skbBlue = app.CreateSeekBar(-1,-1);
skbGreen = app.CreateSeekBar(-1,-1);
skbRed = app.CreateSeekBar(-1,-1);
```

3 MILESTONES

3.1 Milestone 1

Date: Wed Dec 11 19:04:09 2019 -0500
Created a working production version, calling it DroidXTK.

3.2 Milestone 2

Date: Mon Dec 9 15:22:42 2019 -0500
Added android app project + resources - reorganized assets

3.3 Milestone 3

Date: Mon Dec 9 10:24:52 2019 -0500
Wrote initial codes and prototypes.

3.4 Milestone 4

Date: Sat Nov 23 09:12:17 2019 -0500
Created scripts that modify webview through app.Execute() (Executes code in webview in Droidscript)

3.5 Milestone 5

Date: Thu Nov 21 20:45:06 2019 -0500
Added links and repositories for project ideas.

3.6 Milestone 6

Date: Thu Nov 21 11:25:40 2019 -0500
Added final project directory, created symlinks to resources for project potentials, and created link folder to store relevant pages.

3.7 Challenges

Getting voice recognition to work correctly, as it required numerous if/else and switch statements to properly process the code (without it "breaking" the rest of the code).

4 RESULTS

XTK Bridge Method:

```
function dsx()
{
dsx.objectsarray = [];
var objlst = [];
//create new xtk obj counter
this.counter = function (name)
{
//counter name
var cname = "counter_" + name
//counter as new variable (integer counter)
this.cnt = cname;
//set count to 0
cname = 0;
//return counter object
return cname;
}
this.cube = function (name, x, y, z, r, g, b)
{
dsx.setobj(name, x, y, z, r, g, b);
var code = ""
code += (name + " = new X.cube(); \n ")
code += (name + ".center" + " =
[" + x + "," + y + "," + z + "]; \n ")
code += (name + ".color" + " =
[" + r + "," + g + "," + b + " ]; \n ")
code += ("r.add( " + name + "); \n ")
code += ("r.render(); \n ");
return code;
}

...

this.getobjlst = function ()
{
return dsx.objectsarray;
}

...

dsx.writeAsJson = function (path, obj)
{
app.WriteFile(path, JSON.stringify(obj));
}

dsx.readAsJson = function (path)
{
if (app.FileExists(path))
{
return JSON.parse(app.ReadFile(path));
}
app.ShowPopup(path + " does not exist");
return undefined;
}

...
```

5 CONCLUSIONS

Had a great working prototype app, that could create several different geometry types, of which you could track their properties, change them, and create additional ones through use of my main xtk() method. The bridge created through my methods, enabled relatively unbounded access to the Android API's and frameworks, allowing them to exchange data back and forth efficiently.

REFERENCES

[1] **XTK Toolkit:** http://goxtk.com/
[2] **Dave Smart, Droidscript Framework:** http://droidscript.org/