3D Fruit Card documentation

raizensoft

Dec 26, 2024

CONTENTS:

CHAPTER

ONE

GETTING STARTED

1.1 What's inside

- **production**
  Game production ready files
- **dist**
  Compiled and minified Javascript and CSS resources
- **libs**
  External Javascript libraries used in development.
- **src**
  Main javascript source files.
- **test**
  Test folder including data used in development.
- **Gruntfile.js**
  Grunt build file.
- **package.json**
  node.js project file.
- **npm-shrinkwrap.json**
  node.js package dependencies file.
- **docs**
  Documentation includes html and pdf format

1.2 Use production ready files

The *production* folder contains everything you need to deploy the game to different mediums. The files are compressed and optimized to be embedded in website or integrated in a Cordova app.

1.3 Folder Structure

```
assets/
  graphics/
  sounds/
  text/
css/
  fonts/
  fc3d.min.css
  app.css
js/
  fc3d.min.js
data/
index.html
```

- *assets:* Contain all game graphics, sounds and text resources
- *css:* Main stylesheet folder
  - *fonts:* Custom icon fonts folder
  - *fc3d.min.css:* Minimized game stylesheet
  - *app.css:* Generic and non-application specific stylesheet
- *js:*
  - *fc3d.min.js:* Main minimized Javascript file
- *data:* Game additional data
- *index.html:* Main html file

CHAPTER

TWO

CUSTOMIZATION

You can customize the game by editing *config.json* (production/ test folder) or insert javascript code into intializing code in *index.html*

2.1 config.json

- General parameters

Name	Type	Default	Description
useHelpPanel	boolean	true	Enable or disable help panel
useCreditPanel	boolean	true	Enable or disable credit panel
useBackground-Music	boolean	true	Enable or disable background music
cardTextures	array	[“pad1.jpg”, . . .]	List of textures used as card textures

- **String resources for changing text interfaces:**
  APP_TITLE, APP_INFO, NEW_GAME, HELP, HELP_TEXT, SETTING, CREDIT, CREDIT_TEXT
- Input image customization

  The “data” object contains keys and values which correspond to each image category and its number of items.

For example, to add a new category “fruits” you can create a folder “fruits” and put all the images naming from 1.jpg, 2.jpg, 3.jpg... inside that folder. Then declare “data” like so with 20 is the total number of items

```
"data":{
  "fruits":20
}
```

- Level customization

  The “level” object is used to add new game level. In this example “category” is the category id declared in “data” and “grid” is the matrix configuration for that level.

```
"level":[
  {"category":0, "grid":"2x3"},
  {"category":1, "grid":"2x3"},
  {"category":2, "grid":"3x4"},
  {"category":3, "grid":"4x4"}
],
```

2.2 Javascript

You can also pass an argument object to the game instance to config its behaviour. Inside “index.html” (production folder) :

```
var el = document.querySelector('.rs-fc3d');
var fc3d = new FruitCard(el, {
  containerZ:-1400,
  cardWidth:300,
  cardHeight:400,
  cardDistance:50,
  borderScale:1.12,
  fitFactor:0.85,
  ambientLight:0x333333,
  lightMovingSpeed:4
});
```

Name	Type	Default	Description
containerZ	Number	-1400	Container Z position
cardWidth	Number	300	Card item width
cardHeight	Number	400	Card item height
cardDistance	Number	50	Distance between items
borderScale	Number	1.12	White border scaleing
fitFactor	Number	0.96	Fit factor applied to each photo
ambientLight	Number	0x333333	Ambient color of the scene
lightMov-ingSpeed	Number	2	Light moving speed

CHAPTER

THREE

ASSETS

You can reskin the game by replacing many assets in different folders using the same asset names:

```
assets/
  graphics/
  sounds/
  text/
```

- *graphics:* Core game graphics, many can be replaced like logo, padding pattern, board textures...
- *sounds:* Game sound and music effects
- *text:* Help file text content

CHAPTER

FOUR

CORDOVA

Cordova is a mobile development framework that enables developers creates game and app using familiar web technologies. Packaging the game source code to be used in cordova app is really simple.

4.1 Installation

See Cordova - Get Started to install cordova into your system

- Create new Cordova game with a namespace:

```
cordova create MyGame com.mycompany.game
```

- Add “android” platform to deploy the app in android, and “browser” platform to test in browser

```
cordova platform add browser
cordova platform add android
```

- Copy the entire content of *production* folder into *www* folder and overwrite all the files.
- Plug in your android device and test the game using:

```
cordova run android
```

CHAPTER

FIVE

DEVELOPMENT

Using the provided build tools, you can quickly setup the development environment and start customizing the game very easily.

5.1 Setup environment

- Install nodejs
- Run Window command line program (or Terminal app in Linux/Unix OS)
- Change current directory to the game folder
- Type “npm install”
- Once finished, type “grunt” to launch a local http server in the background and start developing
- Test the gallery by go to “http://localhost:3333/test/”

5.2 Asynchronous Module

Modular programming has many advantages over one monolothic code base. All the app components are designed as individual classes wrapped in AMD module and then loaded asynchronously using require.js. This approach ensures the separation of concerns between software components and help implementing future features much easier.

5.3 Entry point

In the main html file, setup an entry Javascript file which configures requirejs and bootstraps the game:

- index.html

```
<script src="js/require.js" data-main="js/entry.js"></script>
```

- entry.js

```
// Setup baseUrl for source folder and library paths
requirejs.config({
  baseUrl:"../src/",
  paths:{
    libs:"../libs/"
  }
});

require(['rs/fc3d/FruitCard', 'libs/domReady'],

  function(FruitCard, domReady) {

    "use strict";

    domReady(function() {

      var el = document.querySelector('.rs-fc3d');
      var fc3d = new FruitCard(el);
      window.fc3d = fc3d;
    });
  });
```

See require.js for more detail about developing with AMD module

CHAPTER

SIX

WORDPRESS INTEGRATION

You can easily insert the game into WordPress post or page using iframe tag

6.1 Upload the game to WordPress

- Inside download package, rename “production” folder to “fruit-card”
- Upload “fruit-card” to WordPress “uploads” folder. The path of “uploads” is “your_site_root/wp-content/uploads”
- Check that you can launch the game by going to “yourwebsite.com/wp-content/uploads/fruit-card/index.html”

6.2 Insert the game into WordPress

- Create a new WordPress post
- Switch to Code Editor by triggering the three dot icon in the top bar and choose Editor - Code editor
- Add some title for the post
- Add iframe tag to embed the game in the post body

```
1  <iframe src="yourwebsite.com/wp-content/uploads/fruit-card/index.html"␣
↪width="1000" height"800"></iframe>
```

- Change the game width and height by modifying those values in iframe tag
- View the post with the game you just embed

CHAPTER

SEVEN

BUILD TASKS

The following build tasks are available for development and production. In the terminal, type `grunt {taskname}` to execute the task

- **default:**
  Default Grunt tasks, this will launch "http-server" and "watch" tasks
- **dist**
  Create distrubitions, including mifnied CSS and Javascript files
- **production**
  Game production ready files
- **http-server**
  Launch a http server in the background for local testing
- **compass**
  Compile SCSS files to CSS files
- **pug**
  Compile pug templates to html files
- **watch**
  Watch for changes in sass and pug folders and auto compile them
- **cssmin**
  Minified application css file, exported in *dist* folder (sdk3d.min.css and sdk3d.light.min.css)
- **clean**
  Various targets to cleanup the project before rebuilding
- **copy**
  Copy assets, css and js files from *test* folder to build examples and dist files
- **requirejs:**
  Build, compile and minify *sdk3d.min.js*, exported in *dist* folder
- **compress:**
  Compressed distributed minified script *sdk3d.min.js* into GZIP format, produce *sdk3d.min.gz.js*