CMPM 120

Text
Objectives

By the end of today you should be able to...

1. **Text & Fonts**
   a. Demonstrate how Phaser displays *displays text*
   b. Demonstrate how to include *custom fonts*
   c. Practice using *git*
      i. Particularly, practice making a *pull request*
Pair up
Fork the repository
Join GitHub today
GitHub is home to over 36 million developers working together to host and review code, manage projects, and build software together.

No description, website, or topics provided.

2 commits 1 branch 0 releases 1 contributor
Branch: master
New pull request

ikarth example javascript for fonts
assets initial commit
Latest commit 90eef17 7 minutes ago
Find File Clone or download
1 hour ago
Download the repository

```
git clone http://github.com/ikarth/phasertext

cd phasertext

git branch yourname

git checkout yourname
```
Why talk about text and fonts?

Most games have some form of text, even just for the score.

Good use of text and fonts makes a huge difference to the look and feel of your game.

Therefore, this will be useful in nearly every game you make, including the ones in this class.
Text Style Properties

**font** string <optional> 'bold 20pt Arial'
The style and size of the font.

**fontStyle** string <optional> (from font)
The style of the font (e.g. 'italic'): overrides the value in `style.font`.

**fontVariant** string <optional> (from font)
The variant of the font (e.g. 'small-caps'): overrides the value in `style.font`.

**fontWeight** string <optional> (from font)
The weight of the font (e.g. 'bold'): overrides the value in `style.font`.

**fontSize** string | number <optional> (from font)
The size of the font (e.g. 32 or '32px'): overrides the value in `style.font`.

**backgroundColor** string <optional> null
A canvas fillstyle that will be used as the background for the whole Text object. Set to `null` to disable.

**fill** string <optional> 'black'
A canvas fillstyle that will be used on the text e.g. 'red', '#00FF00'.

**align** string <optional> 'left'
Horizontal alignment of each line in multiline text. Can be: 'left', 'center' or 'right'. Does not affect single lines of text (see `textBounds` and `boundsAlignH` for that).

**boundsAlignH** string <optional> 'left'
Horizontal alignment of the text within the `textBounds`. Can be: 'left', 'center' or 'right'.

**boundsAlignV** string <optional> 'top'
Vertical alignment of the text within the `textBounds`. Can be: 'top', 'middle' or 'bottom'.

**stroke** string <optional> 'black'
A canvas stroke style that will be used on the text stroke e.g. 'blue', '#FCFF00'.

**strokeThickness** number <optional> 0
A number that represents the thickness of the stroke. Default is 0 (no stroke).

**wordWrap** boolean <optional> false
Indicates if word wrap should be used.

**wordWrapWidth** number <optional> 100
The width in pixels at which text will wrap.

**maxLines** number <optional> 0
The maximum number of lines to be shown for wrapped text.

**tabs** number <optional> 0
The size (in pixels) of the tabs, for when text includes tab characters. 0 disables. Can be an array of varying tab sizes, one per tab stop.
Text Style Properties (continued)

**backgroundColor** string <optional> null
A canvas fillstyle that will be used as the background for the whole Text object. Set to **null** to disable.

**fill** string <optional> 'black'
A canvas fillstyle that will be used on the text eg 'red', '#00FF00'.

**align** string <optional> 'left'
Horizontal alignment of each line in multiline text. Can be: 'left', 'center' or 'right'. Does not affect single lines of text (see textBounds and boundsAlignH for that).

**boundsAlignH** string <optional> 'left'
Horizontal alignment of the text within the textBounds. Can be: 'left', 'center' or 'right'.

**boundsAlignV** string <optional> 'top'
Vertical alignment of the text within the textBounds. Can be: 'top', 'middle' or 'bottom'.

**stroke** string <optional> 'black'
A canvas stroke style that will be used on the text stroke eg 'blue', '#FCFF00'.

**strokeThickness** number <optional> 0
A number that represents the thickness of the stroke. Default is 0 (no stroke).

**wordWrap** boolean <optional> false
Indicates if word wrap should be used.

**wordWrapWidth** number <optional> 100
The width in pixels at which text will wrap.

**maxLines** number <optional> 0
The maximum number of lines to be shown for wrapped text.

**tabs** number <optional> 0
The size (in pixels) of the tabs, for when text includes tab characters. 0 disables. Can be an array of varying tab sizes, one per tab stop.
Text Style Properties (continued)

`boundsAlignV`  string  <optional>  'top'
Vertical alignment of the text within the `textBounds`. Can be: 'top', 'middle' or 'bottom'.

`stroke`  string  <optional>  'black'
A canvas stroke style that will be used on the text stroke eg 'blue', '#FCFF00'.

`strokeThickness`  number  <optional>  0
A number that represents the thickness of the stroke. Default is 0 (no stroke).

`wordWrap`  boolean  <optional>  false
Indicates if word wrap should be used.

`wordWrapWidth`  number  <optional>  100
The width in pixels at which text will wrap.

`maxLines`  number  <optional>  0
Text Style Properties (continued)

**wordWrapWidth**  
number  <optional>  100  
The width in pixels at which text will wrap.

**maxLines**  
number  <optional>  0  
The maximum number of lines to be shown for wrapped text.

**tabs**  
number  <optional>  0  
The size (in pixels) of the tabs, for when text includes tab characters. 0 disables. Can be an array of varying tab sizes, one per tab stop.
Text Style Properties

- **font**: string 'bold 20pt Arial'
- **fontStyle**: string (from font)
- **fontVariant**: string (from font)
- **fontWeight**: string (from font)
- **fontSize**: string | number (from font)
- **backgroundColor**: string null
- **fill**: string 'black'
- **align**: string 'left'
- **boundsAlignH**: string 'left'
- **boundsAlignV**: string 'top'
- **stroke**: string 'black'
- **strokeThickness**: number 0
- **wordWrap**: boolean false
- **wordWrapWidth**: number 100
- **maxLines**: number 0
- **tabs**: number 0

[https://photonstorm.github.io/phaser-ce/Phaser.Text.html](https://photonstorm.github.io/phaser-ce/Phaser.Text.html)
Phaser's Default Style is Cliché and Passé

"bold 20pt Arial"

Both boring and everyone has seen it before.

You can do better!
Phaser.Text extends Phaser.Sprite

What does this mean for us?
Phaser Text as Extended Sprites

Since Text inherits from the Sprite class, we can read and apply sprite-like properties and methods (even weird stuff like damage and health).

That means we can do interesting interactive stuff like grant input control over text or apply physics bodies to text objects.
Custom Fonts
Bitmap Fonts

“BitmapText objects work by taking a texture file and an XML or JSON file that describes the font structure. It then generates a new Sprite object for each letter of the text, proportionally spaced out and aligned to match the font structure.”

Phaser API

Translation: bitmap text is a font that has been laid out in a grid (like a sprite sheet). Bitmap text is less flexible than a text object, but renders much faster.

Atari ST 8x16 System Font
I got this bitmap font from the Phaser examples.

gem.png +

<code>
<svg version="1.1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <image xlink:href="gem.png"/>
</svg>
</code>

<code>
<svg version="1.1" xmlns="http://www.w3.org/2000/svg">
  <image xlink:href="gem.png"/>
</svg>
</code>
// some global variables
var game;

// wait for browser window load, then start the party
window.onload = function() {
    game = new Phaser.Game(650, 650);
    game.state.add('Play', Play);
    game.state.start('Play');
}

var Play = function(game){};
Play.prototype = {
    preload: function() {
        game.load.path = './assets/fonts/';
        game.load.bitmapFont('gem', 'gem.png', 'gem.xml');
    },
    create: function() {
        // bitmapText(x, y, font, text, size, group)
        text01 = game.add.bitmapText(32, 32, 'gem', 'Bitmap text, yeah!', 12);
        text02 = game.add.bitmapText(32, 64, 'gem', 'OK, that was too small... ', 24);
        text03 = game.add.bitmapText(32, 128, 'gem', 'Here\'s the default size of 32 ;p');
        text04 = game.add.bitmapText(32, 256, 'gem', 'NOW I AM SHOUTING!', 64);
        text05 = game.add.bitmapText(32, 375, 'gem', 'BLUR', 256);
    },
    update: function() {
    }
};
What about Google (or other web) fonts?

It’s complicated.

The Phaser examples page has a complex method for using external fonts that involves creating a timer, loading a script file, then using the font in your game. It feels pretty hack-y due to how fonts are loaded online.

But...there's a (slightly, maybe?) easier way.

(Kudos to Arcanorum on html5gamedevs for this technique)
Web Font Loader

Web Font Loader gives you added control when using linked fonts via `@font-face`. It provides a common interface to loading fonts regardless of the source, then adds a standard set of events you may use to control the loading experience. The Web Font Loader is able to load fonts from Google Fonts, Typekit, Fonts.com, and Fontdeck, as well as self-hosted web fonts. It is co-developed by Google and Typekit.

Get Started

To use the Web Font Loader library, just include it in your page and tell it which fonts to load. For example, you could load fonts from Google Fonts using the Web Font Loader hosted on Google Hosted Libraries using the following code.

```html
<script src="https://ajax.googleapis.com/ajax/libs/webfont/1.6.26/webfont.js"></script>

<script>
  WebFont.load({
    google: { families: ['Droid Sans', 'Droid Serif'] }
  });
</script>
```
Web Font Loader

snippet>

<script src="https://ajax.googleapis.com/ajax/libs/webfont/1.6.26/webfont.js"></script>

site:

github.com/typekit/webfontloader

versions:

1.6.26, 1.6.16, 1.5.18, 1.5.10, 1.5.6, 1.5.3, 1.5.2, 1.5.0

Troubleshooting

Seeing an outdated version? Make sure you’re not using the “automatic version” links, like /jqueryui/1/..., but instead use URLs referring to exact versions. Due to concerns over caching and lack of compatibility between even minor versions, we have deprecated and stopped updating the automatic version aliases some time ago, so they will forever refer to an old version (in order to not break existing sites that still use them).

If you encounter problems:

- Look for typos. Remember that JavaScript is a case-sensitive language.
- Use a JavaScript debugger. In Chrome, use the Chrome DevTools. In Firefox, you can use the built-in Firefox DevTools. In IE, you can use the F12 developer tools.
<!DOCTYPE html>
<html>
<head>
<title>Phaser</title>
<link href="https://fonts.googleapis.com/css?family=Pacifico" rel="stylesheet">
<link href="https://fonts.googleapis.com/css?family=Shadows+Into+Light" rel="stylesheet">
<!-- Snag the latest Web Font Loader from Google Hosted Libraries -->
<!-- https://developers.google.com/speed/libraries/#web-font-loader -->
<script src="https://ajax.googleapis.com/ajax/libs/webfont/1.5.18/webfont.js"></script>
<script>
  WebFont.load({
    google: {
      // Load fonts here
      families: ['Pacifico', 'Shadows Into Light']
    }
  });
</script>
<script type="text/javascript" src="../framework/phaser.min.js"></script>
<script type="text/javascript" src="text03.js"></script>
</head>
<body>
</body>
</html>
Pull Requests


Creating a Pull Request:


Merging a Pull Request:

Phaser Examples: Text

https://phaser.io/examples/v2/category/text
Dialog Systems?
What do we need in a dialog system?
Nathan's Architecture

1. Create structured dialog data in JSON
2. Create and position dialog box sprite
3. Check to see if there are dialog lines remaining in current conversation
4. Check to see if there is a new speaker and tween them into view (and tween out previous speaker)
5. Construct dialog by adding speaker + line
6. Create a timer to “fire” dialog letter by letter
7. Lock input until all characters have printed
8. Increment; repeat
Some dialog systems

Yi's PhaserDialog
https://github.com/kthtes/PhaserDialog

April Grow's dialog system

120 student Tina Peng's timed dialog
Other Narrative Tools

Yarn: https://github.com/InfiniteAmmoInc/Yarn

Javascript port of Yarn: https://github.com/jhayley/bondage.js/

Ink (from Inkle): https://www.inklestudios.com/ink/

Javascript port of Ink: https://github.com/y-lohse/inkjs
More Debugging Tips
Useful random debugging advice

1. When you find a problem, change something so that same problem can't happen again
   a. assert()
   b. Keep a debugging notebook
2. Make debug tools
   a. Quicker feedback is better
   b. Display values live if possible
3. Only make one change at a time and then test it
4. Just because you paused the game doesn't mean it's paused
   a. And stopping one update doesn't mean you stopped all of them
5. console.log() is slow
   a. Faster to print an array as a string than to individually print the contents
Useful random debugging advice

Walk through your code step by step, explaining to yourself what is supposed to happen
Useful random debugging advice

1. When you find a problem, change something so that same problem can't happen again
   a. assert()
   b. Keep a debugging notebook

2. Make debug tools
   a. Quicker feedback is better
   b. Display values live if possible

3. Only make one change at a time and then test it

4. Just because you paused the game doesn't mean it's paused
   a. And stopping one update doesn't mean you stopped all of them

5. console.log() is slow
   a. Faster to print an array as a string than to individually print the contents
Useful random debugging advice

1. When you find a problem, change something so that same problem can't happen again
   a. assert()
   b. Keep a debugging notebook
2. Make debug tools
   a. Quicker feedback is better
   b. Display values live if possible
3. Only make one change at a time and then test it
4. Just because you paused the game doesn't mean it's paused
   a. And stopping one update doesn't mean you stopped all of them
5. console.log() is slow
   a. Faster to print an array as a string than to individually print the contents
Useful random debugging advice

1. When you find a problem, change something so that same problem can't happen again
   a. assert()
   b. Keep a debugging notebook
2. Make debug tools
   a. Quicker feedback is better
   b. Display values live if possible
3. Only make one change at a time and then test it
4. Just because you paused the game doesn't mean it's paused
   a. And stopping one update doesn't mean you stopped all of them
5. console.log() is slow
   a. Faster to print an array as a string than to individually print the contents
Useful random debugging advice

1. When you find a problem, change something so that same problem can't happen again
   a. assert()
   b. Keep a debugging notebook

2. Make debug tools
   a. Quicker feedback is better
   b. Display values live if possible

3. Only make one change at a time and then test it

4. Just because you paused the game doesn't mean it's paused
   a. And stopping one update doesn't mean you stopped all of them

5. console.log() is slow
   a. Faster to print an array as a string than to individually print the contents
AABB characters and slopes

An example of a real-world physics-and-debugging problem in a game with 2D physics like yours

https://twitter.com/eevee/status/1133248372624613376