Date: 10-Apr-2023

Vertical parallax

VERSION: 2.1.3

# **Overview**

Parallax scrolling uses multiple objects at different speeds to give the impression of a 3d effect where nearby objects have a larger parallax than more distant objects when observed from different positions. If you're looking for a popular scrolling effect used by native mobile apps and amazing websites, download the vertical parallax.

The Vertical Parallax component is a single-page animated UI that contains a background and a foreground to create a parallax animation.

## **Use case:**

An app where the user wants to display information about a location and an image of the location. Users can use the image as the background of the component and can include the information in the foreground of the component.

## **Percentage of re-use:**

80-90%.

## **Features:**

· Parallax scrolling effect in a vertical manner.

· Ability to animate background and foreground at different speeds

· Facility to customize the UI elements.

# **Getting Started**

## **Prerequisites**

Before you start using the Vertical Parallax component, ensure the following:

• [HCL Foundry](https://manage.hclvoltmx.com/)

• Volt MX Iris

## **Platforms Supported**

### Mobile

#### iOS

#### Android

### PWA

## **Importing the app**

## You can import the Forge components only into the apps that are of the Reference Architecture type.

## **To import the Vertical Parallax component, do the following:**

## Open your app project in Volt MX Iris.

## In the Project Explorer, click the **Templates** tab.

## Graphical user interface, text, application Description automatically generated

## Right-click **Components**, and then select **Import Component**. The **Import Component** dialog box appears.

## Graphical user interface, text, application, Teams Description automatically generated

## Click **Browse** to navigate to the location of the component, select the component, and then click **Import**. The component and its associated widgets and modules are added to your project.

## 

Once you have imported a component to your project, you can easily add the component to a form. For more information, refer Add a Component to a Form

## **Adding widgets to the component**

The Vertical Parallax component uses Target Containers to hold the widgets that you add to the background or the foreground.

You can add widgets or other components to the Target Containers based on your preference. Let us add an Image Widget to the background and a TextArea (TextArea2) Widget to the foreground.

**Adding an Image to the background**

#### 1. From the **Project** explorer, select the **flxBackground** target container.

#### 

#### 2. From the Default Library, drag and drop an Image widget onto the **flxBackground** target container

Graphical user interface, text

Description automatically generated

**Note:**

To fit the image to the background container, you can set the following values for the properties of the Image widget.

You can find these properties in the **Look** tab on the right side of Visualizer.

* **Top**: 0dp
* **Left**: 0dp
* **Width**: 100%
* **Height**: 100%

#### **Adding a Text Area to the foreground**

1. From the **Project** explorer, select the **flxForeground** target container.

## 

## From the **Default Library**, drag and drop a **TextArea** widget onto the **flxForeground** target container.

## Graphical user interface, application Description automatically generated

## **Building and previewing the app**

After performing all the above steps, you can build your app and run it on your device. For more information, you can refer to the Building and Viewing an Application section of the Volt MX User Guide.

You can then run your app to see the Vertical Parallax work in real time.

# **References**

## **Dynamic Usage**

You can also add **Vertical parallax** component dynamically. To do so,

1. In Project **Explorer**, on the **Projects** tab, click **Controllers** section to access the respective **Form Controller**. Create a method and implement the code snippet like the sample code mentioned below.



createComponent: function()

{

/\* Creating the component's object \*/

var VerticalParallax = new com.voltmxmp.verticalparallax(

{

"clipBounds": true,

"height": "100%",

"masterType": constants.MASTER\_TYPE\_USERWIDGET,

"layoutType": voltmx.flex.FREE\_FORM,

"skin": "slForm",

"id": "VerticalParallax",

"isVisible": true,

"left": "0dp",

"top": "0dp",

"width": "100%",

"zIndex": 1

}, {}, {});

/\* Setting the component's properties \*/

VerticalParallax.parallaxSpeed = 5;

VerticalParallax.backgroundHeight = "50%";

VerticalParallax.foregroundHeight = "100%";

VerticalParallax.foregroundTop = "50%";

/\* Setting the Placeholder Properties \*/

VerticalParallax.isFgPlaceholderVisible = true;

VerticalParallax.isImgDetailsVisible = true;

VerticalParallax.isBgPlaceholderVisible = false;

/\*Adding the component to a Form\*/

this.view.add(VerticalParallax);

}

In the code snippet, you can edit the properties of the component as per your requirement. For more information, see Setting Properties.

2. Save the file

## **Properties**

The properties provided on the **Component** tab allow you to customize the elements in the **Vertical parallax** component. These elements can be UI elements, service parameters, and so on. You can set the properties from the Volt MX Iris Properties panel on the right-hand side. You can also configure these properties using JavaScript code.

**General Properties**

**1.Parallax Speed**

|  |  |
| --- | --- |
| **Description:** | Specifies the speed that you want to set for the parallax effect. |
| Syntax: | parallaxSpeed |
| **Type:** | Integer |
| **Read/Write:** | Write |
| **Example:** | this. view. componentID.parallaxSpeed= 5; |

**2. Background Height**

|  |  |
| --- | --- |
| **Description:** | Specifies the height of the background flex container. |
| Syntax: | backgroundHeight |
| **Type:** | String  Constant |
| **Read/Write:** | Read + Write |
| **Default Value:** | “50%” |
| **Example:** | this. view. componentID.backgroundHeight= “50%”; |
| **Remarks:** | * Make sure that the value of this property is equal to the Foreground Top to prevent UI distortions. * For more information on the height of a FlexContainer widget, you can refer to the Height Property in the Quantum Visualizer Widget Programmers Guide. |

**3. Foreground Height**

|  |  |
| --- | --- |
| **Description:** | Specifies the height of the foreground flex container. |
| Syntax: | foregroundHeight |
| **Type:** | String  Constant |
| **Read/Write:** | Read + Write |
| **Default Value:** | “100%” |
| **Example:** | this. view. componentID.foregroundHeight= “100%”; |
| **Remarks:** | For more information on the height of a FlexContainer widget, you can refer to the Height Property in the Quantum Visualizer Widget Programmers Guide. |

**4. Foreground Top**

|  |  |
| --- | --- |
| **Description:** | Specifies the position of the top edge of the foreground flex container. |
| Syntax: | foregroundTop |
| **Type:** | String  Constant |
| **Read/Write:** | Read + Write |
| **Default Value:** | “50%” |
| **Example:** | this. view. componentID.foregroundTop= “50%”; |
| **Remarks:** | * Make sure that the value of this property is equal to the Background Height to prevent UI distortions. * For more information on the top position of a FlexContainer widget, you can refer to the Top Property in the Quantum Visualizer Widget Programmers Guide. |

**2. Placeholder Properties**

**5. Foreground**

|  |  |
| --- | --- |
| **Description:** | Toggles the placeholder text on the foreground flex container. |
| Syntax: | isFgPlaceholderVisible |
| **Type:** | Boolean |
| **Default Value:** | true |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. componentID.isFgplaceholderVisible= “true”; |

**6. image Details**

|  |  |
| --- | --- |
| **Description:** | Toggles the placeholder image on the background flex container. |
| Syntax: | isImgDetailsVisible |
| **Type:** | Boolean |
| **Default Value:** | true |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. componentID.isImgDetailsVisible= “true”; |

**7. Background**

|  |  |
| --- | --- |
| **Description:** | Toggles the placeholder text on the background flex container. |
| Syntax: | isBgPlaceholderVisible |
| **Type:** | Boolean |
| **Default Value:** | true |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. componentID.isBgplaceholderVisible= “false”; |

# **Revision History**

|  |  |
| --- | --- |
| **Asset Version** | **Modifications** |
| Version 2.1.3 | * Bug fixes |
| Version 2.1.2 | * Added the following properties in **General**:   + Background Height   + Foreground Height   + Foreground Top |
| Version 2.1.1 | * Bug fixes and improvements. |
| Version 2.1.0 | * Bug fixes and improvements. |
| Version 2.0.0 | * Updated the UI of the component. * Added the following properties in **General**:   + Parallax Speed * Added the following properties in **Placeholder Properties**:   + Foreground   + Image Details   + Background * Removed the following properties:   + Enable Background Image   + Title Visibility   + Enable Foreground Flex   **Note:** For information on the properties that were removed, you can refer to the [older documentation](https://docs.kony.com/marketplace/VerticalParallax/Content/VerticalParallaxV1.pdf). |
| Version 1.0.0 | * Initial release of asset and documentation. |

## **A. Limitations**

* To avoid UI distortions, make sure that the values for the Background Height and the Foreground Top properties are equal.

## **Known Issues**

* No support for Background Flex animation when a user scrolls down the Foreground. The Android and Windows platforms have the issue.