PhaserUIComponent

Documentation - Phaser 3 UI Framework

2019 Ahmad Arsyel

GET STARTED

Introduction

PhaserUIComponent merupakan framework ekstensi dari Phaser 3 dengan kemampuan membuat komponen User Interface (UI) secara cepat, mudah, portable, dan reusable. Ditulis dalam bahasa Typescript dan memanfaatkan teknologi DOM pada HTML5. Bertujuan agar game developer dapat membuat kontrol dan juga interaksi dalam pengembangan game sesuai keinginan player.

Kenapa PhaserUIComponent?

Saat game developer menemukan bahwa penulisan kode dan pembuatan komponen UI yang telah dibuat, harus dibuat lagi dengan algoritma yang sama pada proyek lain. PhaserUIComponent akan menangani hal tersebut, sehingga developer tidak perlu lagi mendesain dan membangun satuan komponen UI kembali untuk setiap proyek.

Apa saja kemampuannya?

Sampai saat ini, terdapat tiga komponen utama yang difasilitasi oleh PhaserUIComponent, yaitu Button, Virtual Control, dan TextField. Untuk pengembangan kedepannya, PhaserUIComponent dapat menambahkan komponen UI lainnya untuk mendukung Phaser 3.

Bahasa yang mendukung

Bahasa utama yang didukung framework PhaserUIComponent adalah Typescript. Dapat menggunakan JavaScript, namun dibutuhkan sedikit konfigurasi dan penyesuaian untuk dapat digunakan.

Lisensi

Ahmad Arsyel - ReydVires

You/Your

Means any person who would like to copy, distribute, or modify the Framework.

Framework

Means the collection of files distributed by the Copyright Holder, and derivatives of that collection and/or of those files. A given Framework may consist of either the Standard Version, or a Modified Version.

Distributed

Means providing a copy of the Framework or making it accessible to anyone else, or in the case of a company or organization, to others outside of your company or organization.

Standard Version

Refers to the Framework if it has not been modified, or has been modified only in ways explicitly requested by the Copyright Holder.

Modified Version

Means the Framework, if it has been changed, and such changes were not explicitly requested by the Copyright Holder.

License

You are permitted to use the Standard Version and create and use Modified Versions for any purpose without restriction, provided that you do not Distribute the Modified Version.

You may Distribute verbatim copies of the Source form of the Standard Version of this Framework in any medium without restriction, either free or for a Distributor Fee, provided that you duplicate all of the original copyright notices. At your discretion, such verbatim copies may or may not include a Compiled form of the Framework.

Any use, modification, and distribution of the Standard or Modified Versions is governed by this Artistic License. By using, modifying or distributing the Framework, you accept this license. Do not use, modify, or distribute the Framework, if you do not accept this license.

This license does not grant you the right to use any trademark, service mark, tradename, or logo of the Copyright Holder.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Installing PhaserUIComponent

Installation

Prequisite

- Pastikan Node.js dan Phaser 3 telah ter-install pada komputer Anda
- Disarankan untuk menggunakan VSCode sebagai IDE

- Sangat disarankan menggunakan bundler Webpack dalam menjalankan proyek (Link template project official tersedia)
- Disarankan untuk proyek Phaser 3 dengan Typescript

Install

- Download Framework PhaserUIComponent [https://s.id/phaseruicomponent]
- Lakukan extract pada file zip
- Pindahkan folder components ke folder src pada proyek Phaser 3 target
- PhaserUIComponent siap digunakan

Template Project GitHub (Typescript)

Terdapat dua tempate project yang dapat digunakan, yaitu dengan sudah terpasang framework PhaserUIComponent, dan yang belum.

1. Default Template Project [Link here!]
2. PhaserUIComponent Template Project [Link here!]

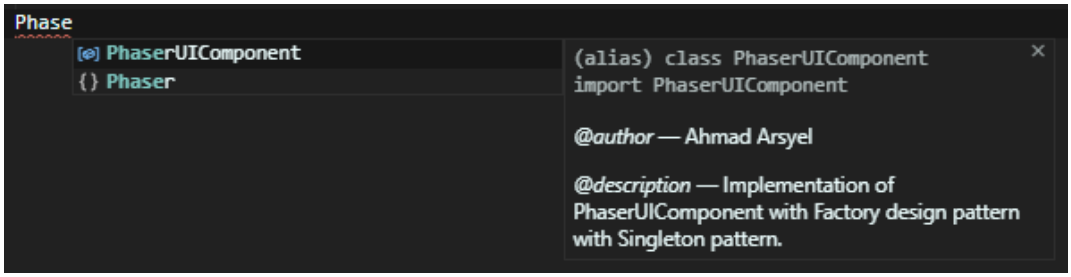
Import to Current Project

1. Lakukan copy folder component PhaserUIComponent Template Project, kemudian pindahkan ke folder src target
2. Atau component bisa didapatkan dari link download Framework PhaserUIComponent, kemudian pindahkan ke folder src target

Component Usage

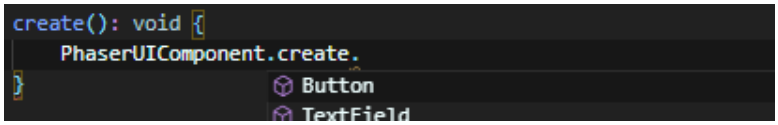
Konsep pemanggilan

Untuk memastikan PhaserUIComponent telah ter-install dengan benar pada proyek Anda, lakukan pemanggilan fungsi menggunakan syntax `PhaserUIComponent`. Jika muncul keterangan autocomplete seperti Gambar 1, maka PhaserUIComponent telah terpasang.



Gambar 1. Tampilan pemangginal syntax PhaserUIComponent.

PhaserUIComponent merupakan komponen yang dibangun dengan metode design pattern gabungan dari singleton dan factory method. Gunakan syntax `PhaserUIComponent.create.` untuk memanggil komponen yang tersedia. Perhatikan Gambar 2.



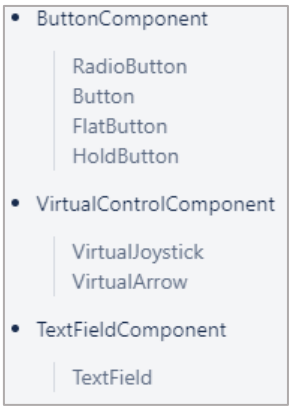
Gambar 2. Syntax utama memanggil komponen UI.

Kelas komponen

Terdapat tiga tipe komponen utama yang dapat dipilih, yaitu Button, VirtualControl, dan TextField. Saat hendak menggunakan salah satu komponen UI tersebut, gunakanlah tipe data yang memiliki sufix `-Component`, yaitu:

- `ButtonComponent`
- `VirtualControlComponent`
- `TextFieldComponent`

Setiap tipe komponen memiliki beberapa implementasi konkrit yang disembunyikan. Perhatikan Gambar 3 untuk detailnya.



Gambar 3. Breakdown UI component.

- `RadioButton`: Membuat tombol dengan fungsi mengembalikan nilai on/off
- `Button`: Membuat tombol dengan mengandalkan dua sprite frame ketika ditekan dan dilepas
- `FlatButton`: Membuat tombol sederhana, dengan efek menekan ketika dilepas
- `HoldButton`: Membuat tombol yang merekam state tiap framenya saat ditekan, dan memberi kondisi saat tombol dilepas, dengan efek sedikit transparan
- `VirtualJoystick`: Membuat Joystick pada game, dengan konfigurasi yang dapat disesuaikan
- `VirtualArrow`: Membuat virtual control D-Pad atau sejenisnya, konfigurasi arah dapat disesuaikan
- `TextField`: Membuat input text pada game, style dapat disesuaikan. Menggunakan DOM dalam implementasinya.

Note: Anda tidak dapat langsung memilih atau mengimplementasikan kelas konkrit didalamnya, atau akan kesulitan dalam melakukan konfigurasi.

API DOCUMENTATION

PhaserUIComponent as Singleton

Cara pemanggilan umum

```
const btnConfig = <PhaserUIComponent.Button.Config> {
    type: 'Button',
    spritesheetTexture: 'btn_ui',
    callback: () => console.log("Hello!")
};
const playButton = PhaserUIComponent.create.Button(this, 180, 320, btnConfig);
```

Gambar 4. Contoh pembuatan Button dengan spritesheetTexture.

Perhatikan Gambar 4. Bentuk umum syntax yang digunakan untuk membuat komponen UI adalah

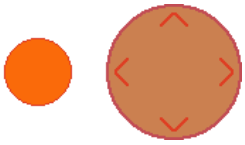
`PhaserUIComponent.create.[component].([scene], [pos X], [pos Y], [config], [optional]);`

- `[component]` ➔ Jenis kelas komponen yang akan dibuat
- `[scene]` ➔ Tempat atau lokasi scene untuk komponen yang akan dibuat
- `[pos X]` ➔ Posisi sumbu X komponen pada scene
- `[pos Y]` ➔ Posisi sumbu Y komponen pada scene
- `[config]` ➔ Untuk setiap tipe komponen yang dipilih memiliki konfigurasinya masing-masing
- `[optional]` ➔ Beberapa tipe komponen memiliki opsi tambahan pada parameter untuk pengaturan komponen

Aturan Tambahan

Gambar 5. Contoh sprite untuk parameter spritesheetTexture (atas onpinterup, bawah onpointerdown) di Button.

Pada contoh asset (assets/btn_ui.png) yang ada pada tempate proyek PhaserUIComponent, komponen Button untuk format spritesheetTexture adalah seperti pada Gambar 5. Spritesheet asset dapat dipecah, dan disesuaikan pada parameter `[config]`.



Gambar 6. Contoh sprite untuk parameter spritesheetTexture (kiri controller, kanan base) di Joystick.

Pada contoh asset (assets/virtualjoystick.png) yang ada pada tempate proyek PhaserUIComponent, komponen VirtualJoystick untuk format spritesheetTexture adalah seperti pada Gambar 6. Spritesheet asset dapat dipecah, dan disesuaikan pada parameter `[config]`.

Component Class

Class ButtonComponent

Syntax API yang digunakan:

```
PhaserUIComponent.create.Button.([scene], [pos X], [pos Y], [config]);
```

- `[scene]: Phaser.Scene`
  Tempat atau lokasi scene untuk ButtonComponent yang akan dibuat
- `[pos X]: number`
  Posisi sumbu X komponen pada scene
- `[pos Y]: number`
  Posisi sumbu Y komponen pada scene
- `[config]: PhaserUIComponent.Button.Config`
  Untuk konfigurasi ButtonComponent

```
PhaserUIComponent.Button.Config
```

- `type?: string`
  Tipe tombol yang akan dibuat berdasarkan ButtonComponent, dengan tipe: Flat (default), Hold, Radio, Button
- `spritesheetTexture?: string`
  Tekstur untuk tombol berdasarkan spritesheet 2 frame
- `texture?: string`
  Tekstur untuk tombol
- `label?: string`
  Teks untuk tombol
- `style?: Phaser.Types.GameObjects.Text.TextStyle`
  Pemberian style untuk teks pada tombol
- `callback?: Function`
  Fungsi yang akan dipanggil setelah aksi tombol dilakukan
- `arg?: any`
  Parameter atau argumen untuk tombol
- `callbackOnUp?: boolean`
  Fungsi `callback` akan dipanggil juga saat tompol dilepas
- `onToggleTexture?: string`
  Tekstur yang akan dijalankan untuk tipe tombol Radio saat ditekan
- `isToggleActive?: boolean`
  Hanya berfungsi untuk tipe tombol Radio, mengatur kondisi awal tombol

Method

```
setCallback(callback: Function, arg?: any): this
```

⇨ `callback`: Fungsi yang akan dipanggil setelah aksi tombol dilakukan
⇨ arg?: Parameter atau argumen untuk tombol
⇨ return ➔ ButtonComponent

```
deactive(): boolean
```

⇨ `properties`

⇨ return ➔ boolean

`deactive(value: boolean)`

⇨ properties

`isPressed(): boolean`

⇨ return ➔ boolean

Class VirtualControlComponent

Syntax API yang digunakan:

```
PhaserUIComponent.create.VirtualControl([scene], [pos X], [pos Y], [config]);
```

- [scene]: `Phaser.Scene`
  Tempat atau lokasi scene untuk VirtualControlComponent yang akan dibuat
- [pos X]: `number`
  Posisi sumbu X komponen pada scene
- [pos Y]: `number`
  Posisi sumbu Y komponen pada scene
- [config]: `PhaserUIComponent.Virtual.Control.ObjectConfig`
  Untuk konfigurasi VirtualControlComponent

`PhaserUIComponent.Virtual.Control.ObjectConfig`

- `type?: string`
  Tipe kontrol yang akan dibuat berdasarkan VirtualControlComponent, dengan tipe: Joystick (default), Arrow atau DPAD
- `spritesheetTexture?: string`
  Tekstur untuk kontrol berdasarkan spritesheet 2 frame
- `texture?: string`
  Tekstur untuk kontroler tengah Joystick
- `container?: string`
  Tekstur untuk background joystick
- `controlled?: object`
  Objek yang hendak disimulasikan oleh joystick
- `isPhysics?: boolean`
  Mengambil physics body dari object untuk dikendalikan
- `width?: number`
  Hanya berfungsi untuk tipe Arrow atau DPAD, mengatur padding lebar tombol
- `height?: number`
  Hanya berfungsi untuk tipe Arrow atau DPAD, mengatur padding tinggi tombol
- `callback?: Function`
  Kustomisasi fungsi untuk tombol dengan atribut tambahan (dalam argumen), yaitu arrowKey pada kontrol tipe Arrow atau DPAD, berfungsi untuk mendetekasi kondisi tombol arah yang sedang ditekan.
- `argument?: any`
  Mengembalikan nilai argumen yang diberikan. Jika tipe Arrow atau DPAD, akan ada objek tambahan yaitu arrowKey, dengan keterangan RIGHT, LEFT, UP dan DOWN
- `arrowConfig?: PhaserUIComponent.Virtual.Control.ArrowConfig`

Konfigurasi detail pemetaan arrow yang akan digunakan

`PhaserUIComponent.Virtual.Control.ArrowConfig`

- `down?: boolean`
  Mengatur ketersediaan tombol bawah
- `left?: boolean`
  Mengatur ketersediaan tombol kiri
- `up?: boolean`
  Mengatur ketersediaan tombol atas
- `right?: boolean`
  Mengatur ketersediaan tombol kanan

Method

`enableControl(active?: boolean): this`

- ⇨ `active?:` Mengatur aktif kontrol
- ⇨ `return` ➔ VirtualControlComponent

`setControlled(controlledObject: any, isPhysics?: boolean): this`

- ⇨ `controlledObject:` Objek yang dikontrol, harus memiliki atribut koordinat x dan y
- ⇨ `isPhysics?:` Kontrol objek dengan body
- ⇨ `return` ➔ VirtualControlComponent

`setOriginalPos(x: number, y: number): this`

- ⇨ `x:` Mengatur koordinat sumbu x
- ⇨ `y:` Mengatur koordinat sumbu y
- ⇨ `return` ➔ VirtualControlComponent

`setControlSpeed(value: number): this`

- ⇨ `value:` Megatur nilai kecepatan
- ⇨ `return` ➔ VirtualControlComponent

Class TextFieldComponent

Syintax API yang digunakan:

`PhaserUIComponent.create.TextField([scene], [pos X], [pos Y], [config] [, style]);`

- `[scene]: Phaser.Scene`
  Tempat atau lokasi scene untuk TextFieldlComponent yang akan dibuat
- `[pos X]: number`
  Posisi sumbu X komponen pada scene
- `[pos Y]: number`
  Posisi sumbu Y komponen pada scene
- `[config]: PhaserUIComponent.TextField.Config`
  Untuk konfigurasi TextFieldComponent
- `[, style]: PhaserUIComponent.TextField.Style`
  Opsional untuk konfigurasi style pada TextFieldComponent

`PhaserUIComponent.TextField.Config`

- `id: string`
  Unik ID untuk komponen textfield pada DOM
- `texture?: string`
  Kustomisasi texture untuk placeholder background textfield
- `placeholder?: string`
  Teks placeholder untuk input
- `inputWidth?: string`
  Atur lebar textfield berdasarkan banyaknya karakter
- `type?: string`
  Tipe textfield, Flat (default)

`PhaserUIComponent.TextField.Style`

- `height?: number`
  Mengatur tinggi texfield dari dalam px
- `padding?: number`
  Mengatur nilai padding pada textfield
- `bgColor?: string`
  Mengatur warna background pada textfield
- `fontSize?: number`
  Mengatur ukuran font pada textfield
- `textAlign?: string`
  Mengatur jajaran teks pada textfield
- `textColor?: string`
  Mengatur warna font pada textfield

Method

`setStyle(style: PhaserUIComponent.TextField.Style): this`

- ⇨ `style:` Mengatur style dari komponen textfield
- ⇨ `return` ➔ TextFielComponent

`resetValue(): this`

- ⇨ Mengosongkan value dari textfield
- ⇨ `return` ➔ TextFielComponent

`getValue(): string`

- ⇨ `return` ➔ Mendapatkan value dari textfield

`onFocus(): boolean`

- ⇨ `return` ➔ Apakah komponen elemen sedang terpilih

`setPosition(x: number, y: number, z?: number, w?: number): this`

- ⇨ `x:` Mengatur nilai posisi komponen di sumbu x
- ⇨ `y:` Mengatur nilai posisi komponen di sumbu y
- ⇨ `z?:` Mengatur nilai posisi komponen di sumbu z
- ⇨ `w?:` Mengatur nilai posisi komponen di sumbu w

- ⇨ `return` ➔ TextFieldComponent

`setX(value: number): this`

- ⇨ `value`: Mengatur nilai posisi komponen di sumbu x
- ⇨ `return` ➔ TextFieldComponent

`setY(value: number): this`

- ⇨ `value`: Mengatur nilai posisi komponen di sumbu y
- ⇨ `return` ➔ TextFieldComponent

`destroy(): any`

- ⇨ `return` ➔ Data objek

`setVisible(value: boolean): this`

- ⇨ `value`: Mengatur nilai tampak objek untuk di-render
- ⇨ `return` ➔ TextFieldComponent

EXAMPLES

Pada template project PhaserUIComponent, example penggunaan komponen sudah tersedia pada ExampleScene.

ButtonComponent

- FlatButton

```
const flatButton = PhaserUIComponent.create.Button(this, 180, 380, {
	texture: 'btn_flat',
	callback: () => console.log("Test FlatButton")
});
```

- Button

```
const btnConfig = <PhaserUIComponent.Button.Config> {
	type: 'Button',
	spritesheetTexture: 'btn_ui',
	callback: () => console.log("Test Button")
};
const button = PhaserUIComponent.create.Button(this, 180, 300, btnConfig);
```

- HoldButton

```
const holdButton = PhaserUIComponent.create.Button(this, 180, 220, {
	type: 'Hold',
	texture: 'btn_flat',
	callback: () => console.log("Test HoldButton"),
	callbackOnUp: true
});
```

- RadioButton

```
const radioButton = PhaserUIComponent.create.Button(this, 180, 140, {
	type: 'Radio',
	texture: 'btn_flat',
	onToggleTexture: 'btn_flat_on',
	isToggleActive: false,
	callback: (isOn: boolean) => console.log("Test RadioButton", isOn)
});
```

VirtualControlComponent

- VirtualJoystick

```
const joystickConfig = <PhaserUIComponent.Virtual.Control.ObjectConfig> {
	type: 'Joystick',
	spritesheetTexture: 'vj',
	controlled: this.player
};
const joystick = PhaserUIComponent.create.VirtualControl(this, 90, 520, joystickConfig);
```

- VirtualArrow

```
const arowConfig = <PhaserUIComponent.Virtual.Control.ArrowConfig> {
	down: false,
};
const dpadConfig = <PhaserUIComponent.Virtual.Control.ObjectConfig> {
	type: 'DPAD',
```

```
        spritesheetTexture: 'btn_arrow',
        arrowConfig: arowConfig,
        width: 72,
        height: 72
};
const dpad = PhaserUIComponent.create.VirtualControl(this, 255, 535,
dpadConfig);
```

TextFieldComponent

- TextField

```
// create()
const tfConfig = <PhaserUIComponent.TextField.Config> {
        id: 'name',
        placeholder: 'Input Name',
};
this.textField = PhaserUIComponent.create.TextField(this, 180, 16,
tfConfig);

// update()
if (this.textField.onFocus()) {
        this.input.keyboard.disableGlobalCapture();
}
const enterKey = this.input.keyboard.addKey('ENTER');
if (Phaser.Input.Keyboard.JustDown(enterKey) && this.textField.onFocus()) {
        if (this.textField.getValue().length > 0) {
                console.log(this.textField.getValue());
                this.textField.resetValue();
        }
}
```