

Table of Contents

Intermediate 2

| | | |
|--|--|----|
| Meeting 9 | | |
| Corona Buster : Move Player with Animation..... | | 1 |
| Meeting 10 | | |
| Corona Buster : Create a New Method..... | | 8 |
| Meeting 11 | | |
| Corona Buster : Kill The Virus..... | | 19 |
| Meeting 12 | | |
| Corona Buster : Add Score Label..... | | 29 |
| Meeting 13 | | |
| Corona Buster : Decrease Life..... | | 39 |
| Meeting 14 | | |
| Corona Buster : Increase Life..... | | 49 |
| Meeting 15 | | |
| Intermediate 2 Test..... | | 59 |
| Meeting 16 | | |
| Intermediate 2 Test..... | | 60 |

Intermediate 3

| | | |
|--|--|-----|
| Meeting 1 | | |
| Math Fighter : Setup The Animation..... | | 62 |
| Meeting 2 | | |
| Math Fighter : Start Game..... | | 82 |
| Meeting 3 | | |
| Math Fighter : Add Number and Question..... | | 95 |
| Meeting 4 | | |
| Math Fighter : Sprite Hit..... | | 108 |
| Meeting 5 | | |
| Math Fighter : Score and Timer..... | | 117 |
| Meeting 6 | | |
| Memory Game : Adding Boxes..... | | 126 |
| Meeting 7 | | |
| Memory Game : Open Box..... | | 141 |

Table of Contents

Meeting 8

Memory Game : Check for Match..... **156**

Meeting 9

Memory Game : Countdown Timer..... **170**

Meeting 10

Make Your Own Game : Exploring Idea..... **177**

Meeting 11

Make Your Own Game : Start Design..... **186**

Meeting 12

Make Your Own Game : Code The Logic..... **193**

Meeting 13

Make Your Own Game : Let's Continue..... **202**

Meeting 14

Make Your Own Game : Debugging..... **209**

Meeting 15

Make Your Own Game : Finishing and Testing..... **216**

Meeting 16

Make Your Own Game : Deploy and Share..... **221**



Timedoctor
Coding Academy



Timedoctor
Coding Academy



CORONA BUSTER : MOVE PLAYER WITH ANIMATION



Apa Yang Akan Kita Pelajari?

1. Menambahkan Player
2. Menggerakkan Player
3. Membuat Animasi Player



Pertemuan

q

Corona Buster



Warm Up!



Apa saja yang sudah kamu tambahkan pada game mu di meeting sebelumnya?

Apa konsep yang digunakan untuk menggerakkan Clouds?

Nah pada pertemuan ini kita akan melanjutkan game Corona Buster dengan **menambahkan Player dan membuatnya bisa bergerak ke kanan dan ke kiri**.

Selain itu kita juga akan **menambahkan animasi pada Player**.

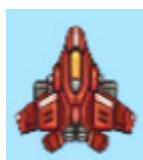
Sama halnya dengan game Bunny Jump, kita membuat Player berganti animasi saat melompat dan jatuh.



Player Ship akan berganti animasi saat bergerak ke kanan dan ke kiri **menggunakan sprite sheet yang sudah tersedia!**



Left



Stand By



Right



Langkah pertama adalah tambahkan Player terlebih dahulu.



Menambahkan Player



Ship adalah image yang berbentuk file spritesheet.
Untuk menambahkan player ship ini, ayo kita lakukan inisialisasi properti, dan juga load spritesheet player.

- ## 1. Inisialisasi variabel global di method init()

```
this.player = undefined
```

- ## 2. Load spritesheet

```
this.load.spritesheet('player', 'images/ship.png',
{frameWidth: 66, frameHeight: 66})
```

Selain itu kita akan menggunakan variabel speed untuk player

- ### **3** Inisialisasi variabel speed di method init()

```
this.speed = 100
```



Pertemuan 9-Corona Buster: Move Player With Animation

Player Animation

Sekarang kita akan membuat animasi untuk player saat digerakkan. Caranya sama dengan saat kamu membuat player Collecting Star loh!



Codes

```
createPlayer ()  
{  
    const player = this.physics.add.sprite(200, 450,  
        'player')  
    player.setCollideWorldBounds(true)  
  
    this.anims.create({  
        key: 'turn',  
        frames:[ { key: 'player', frame: 0 } ],  
    })  
  
    this.anims.create({  
        key: 'left',  
        frames: this.anims.generateFrameNumbers(  
            'player', {start: 1, end: 2}),  
        frameRate: 10  
    })  
  
    this.anims.create({  
        key: 'right',  
        frames: this.anims.generateFrameNumbers('player',  
            {start:1, end:2}),  
        frameRate: 10  
    })  
  
    return player  
}
```



Selanjutnya, panggil method createPlayer() untuk memunculkan player



Codes

```
this.player = this.createPlayer()
```

Coba reload game mu dan pastikan player Ship sudah muncul pada layout.

Nah selanjutnya kita akan membuat method baru bernama movePlayer() untuk membuat player dapat bergerak.

Tulis kode ini di bawah method createButton()



Codes

```
movePlayer(player)
{
    if (this.nav_left){
        this.player.setVelocityX(this.speed * -1)
        this.player.anims.play('left', true)
        this.player.setFlipX(false)
    } else if (this.nav_right){
        this.player.setVelocityX(this.speed)
        this.player.anims.play('right', true)
        this.player.setFlipX(true)
    } else {
        this.player.setVelocityX(0)
        this.player.anims.play('turn')
    }
}
```

Selesai!

Ayo coba gerakkan player ship di game mu!



Pertemuan 8-Corona Buster: Add Button and Player Adding Clouds

Selanjutnya, panggil method movePlayer().
Tulis kode ini pada method update()



Codes

```
this.movePlayer(this.player)
```



Let's Explore

Explore 1

Masih ingatkah kamu cara menggerakkan player menggunakan key arrow pada keyboard?
Ayo latih pemahamanmu kembali!

Cara menggerakkan player dengan key arrow

Petunjuk :

1. Inisialisasi properti cursor pada method init() dengan nilai undefined
2. Tambahkan cursor pada method create()
3. Tambahkan pada method movePlayer() di dalam kondisi if else

Contoh:

```
if (this.cursors.left.isDown || this.nav_left)
```

Kamu bisa melihat game yang sebelumnya agar lebih mudah ya!

Explore 2

Tambahkan kondisi baru untuk menggerakkan Player ke atas dan ke bawah dan berikan animasi 'turn'!

Explore 3

Berikan efek suara saat Player berpindah!

Kamu bisa mencari efek suara di link ini :

<https://www.soundsnap.com/tags/spaceship>

<https://www.storyblocks.com/audio>



Hari/Tanggal : _____

1. Bagaimana kode untuk menambahkan Player pada method createPlayer() ?

.....
.....
.....

2. Player memiliki animasi saat digerakkan, sebutkan animasi apa saja yang dibuat!

.....
.....

3. Coba tuliskan contoh kode untuk membuat animasi Player!

.....
.....
.....

4. Method apa yang dibuat untuk menggerakkan Player?

.....
.....

5. Apa arti kode ini yang ditulis pada method movePlayer() ?

`this.player.setVelocityX(this.speed * -1)`

.....
.....
.....



Timedoctor
Coding Academy

10

CORONA BUSTER : CREATE A NEW METHOD



Apa Yang Akan Kita Pelajari?

1. Membuat kelas falling object
2. Menambahkan enemy



Corona Buster

Ayo kita lanjutkan game di pertemuan sebelumnya!
Kita akan menambahkan **Enemy** yaitu virus Corona Bagaimana cara membuatnya?
Ikuti langkah-langkah ini ya!



Menambahkan Kelas Falling Object

Kelas Falling Object digunakan untuk **menambahkan konfigurasi dan method dari objek yang menggunakan kelas Falling Object.**

Nantinya kelas ini akan digunakan oleh **Enemy** dan juga **Handsantizer**.

Terdapat beberapa method yang akan dibuat pada kelas Falling Object.

Nah ikuti langkah berikut ini untuk membuat kelas Falling Object ya!



Buat folder baru **pada src** bernama **ui** (boleh dengan nama yang lain).
Lalu **pada folder itu**, buat file baru bernama **FallingObject**.

Codes

```
import Phaser from 'phaser'

export default class FallingObject extends Phaser.Physics.Arcade.Sprite
{
    constructor(scene, x, y, texture, config)
    {
        super(scene, x, y, texture)

        this.scene = scene
        this.speed = config.speed
        this.rotationVal = config.rotation
    }
}
```



Ingat dengan kelas Carrot yang pernah kita buat pada game Bunny Jump?

Sama seperti kelas Carrot, kelas Falling Object ini juga menggunakan constructor dengan parameter scene, x, y, dan texture.

Tambahannya, kita juga menggunakan parameter config dari kelas GameConfig yang disediakan oleh phaser untuk mengakses speed dan rotation.

Nah sekarang kita akan menambahkan method baru pada kelas Falling Object yaitu **spawn()** dan **die()**

Method ini nantinya dapat dipanggil pada CoronaBusterScene.



Pertemuan 10-Corona Buster: Create a New Method Falling Object Class

Lanjutkan kode di bawah ini di bawah method constructor.



Codes

```
import Phaser from 'phaser'

export default class FallingObject extends Phaser.Physics.Arcade.Sprite
{
    //lanjutkan dari kode ini
    spawn(x) {
        const positionY = Phaser.Math.Between(-50, -70)

        this.setPosition(y, positionX)

        this.setActive(true)
        this.setVisible(true)
    }

    die() {
        this.destroy()
    }
}
```

Method spawn() -> untuk memunculkan objek enemy

Method die() -> untuk menghancurkan objek enemy

Nah kita masih perlu satu method lagi di dalam kelas Falling Object yaitu method **update()**. Fungsi method update() disini juga sama dengan method update() pada scene yaitu untuk merekam perubahan yang terjadi secara terus menerus **(forever)**



Codes

```
import Phaser from 'phaser'

export default class FallingObject extends Phaser.Physics.Arcade.Sprite
{
    //lanjutkan dari kode ini
    update(time) {
        this.setVelocityY(this.speed)
        this.rotation += this.rotationVal
        const gameHeight = this.scene.scale.height

        if (this.y > gameHeight + 5) {
            this.die()
        }
    }
}
```



Kenapa method update() berisi parameter time?
Apa fungsi dari parameter time?



Vocabulary

Parameter **Time** berfungsi untuk menyimpan waktu sekarang (**current time**) dan digunakan ketika game ingin menampilkan objek berdasarkan rentang waktu tertentu.

Jika di scratch kita bisa langsung menggunakan kode wait 1 second.
Tapi di Phaser, kita **perlu menambahkan parameter Time jika ingin menggunakan kode wait (delay)**.

wait 1 seconds

Jadi pada game ini, child dari Falling Object akan dimunculkan pada rentang waktu tertentu.



Nah kelas Falling Object sudah selesai dibuat. Ayo sekarang kita kembali ke CoronaBusterScene!



Menambahkan Enemy

Sebelum menambahkan enemy, kita import kelas Falling Object dulu karena Enemy akan menggunakan konfigurasi dari kelas Falling Object.

```
import FallingObject from '../ui/FallingObject.js'
```

Kemudian kita inisialisasi properti enemy pada method init(). Selain enemy, inisialisasi juga properti **speed** karena kita akan membutuhkannya nanti



Codes

```
// di dalam method init()  
this.enemies = undefined  
this.enemySpeed = 60
```

Selanjutnya load image enemy.

Coba kamu load image enemy mu sendiri ya!

Caranya sama dengan load image biasa loh!



Sekarang, untuk menambahkan enemy kita akan memerlukan method baru yang diberi nama **spawnEnemy()**



1. Membuat Method spawnEnemy()

Untuk memunculkan enemy, kita akan memerlukan properti **speed** dan **rotation** yang disediakan oleh phaser.



Vocabulary

Speed : mengatur kecepatan bergeraknya objek

Rotation : mengatur adalah arah perputaran objek dan kecepatan berputarnya. Nilai positif artinya berputar ke kanan dan negatif artinya berputar ke kiri.

Nah sekarang buat method **spawnEnemy()** di bawah method **movePlayer()**



Codes

```
spawnEnemy()
{
    const config = {
        speed : this.enemySpeed,
        rotation : 0.06
    }

    const enemy = this.enemies.get(0, 0, 'enemy', config)

    const enemyWidth = enemy.displayWidth

    const positionX = Phaser.Math.Between(enemyWidth,
        this.scale.width - enemyWidth)

    if (enemy) {
        enemy.spawn(positionX)
    }
}
```



Challenge!

1. Coba ubah nilai speed menjadi berapapun sesukamu
2. Coba ubah nilai rotation menjadi nilai ini secara bergantian
 - rotation = 0
 - rotation = -1
 - rotation = 0.1

Selanjutnya kita akan menambahkan enemy. Enemy akan menjadi objek grup yang dinamis (dapat bergerak) sehingga kita akan menggunakan physics group.



2. Menambahkan Enemy

Enemy yang kita buat termasuk ke dalam kelas FallingObject, artinya Enemy dapat menggunakan method yang sudah dibuat pada kelas FallingObject.

Pada method `create()`, tambahkan kode ini.



Codes

```
this.enemies = this.physics.add.group({
    classType : FallingObject,
    //banyaknya enemy dalam satu kali grup
    maxSize : 10,
    runChildUpdate : true
})
```



Vocabulary

runChildUpdate hanya bernilai **true** atau **false**.

Fungsinya untuk memanggil method update yang terdapat pada kelas FallingObject.



2. Menambahkan Timer Event

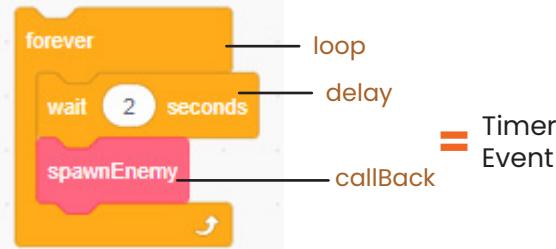
Nah selanjutnya kita akan membuat enemy tidak muncul secara bersamaan, tapi ada jarak waktu tertentu. Kita akan menggunakan kelas baru dari Phaser yaitu Timer Event.



Timer Event berasal dari kelas Phaser.Time.TimerEvent yang berfungsi untuk membuat jarak waktu sebelum melakukan method selanjutnya.

Timer Event bisa berisi parameter delay, callBack, callbackScope, loop, tick, repeatCounts, dan argument.

Jika di Scratch, kita bisa **selamanya memanggil function spawnEnemy dengan jarak 2 detik** menggunakan kode di samping.



Sekarang kita tambahkan Timer Event untuk method spawnEnemy()

Lanjutkan kode ini di bawah kode yang baru saja kamu buat tadi pada method `create()`.



Codes

```
this.time.addEvent({
  delay: 2000,
  callback: this.spawnEnemy,
  callbackScope: this,
  loop: true
})
```



Pertemuan 10-Corona Buster: Create A New Method Adding Enemy

Nah selesai! Coba jalankan game mu dan pastikan enemy sudah berhasil dimunculkan ya!



Challenge!

Masih ingat kode untuk mengambil nilai acak (random) kan?
Coba kamu ubah nilai
delay: 2000
menjadi nilai acak antara 2 detik sampai 8 detik



Hari/Tanggal : _____

1. Objek apa saja yang menggunakan kelas FallingObject?

.....
.....
.....

2. Apa fungsi method spawn() dan die() pada kelas FallingObject?

.....
.....
.....

3. Apa fungsi parameter time pada method update()?

.....
.....
.....

4. Apa yang terjadi jika rotation bernilai negatif?

.....
.....
.....

5. Apa itu Timer Event? Bagaimana contoh Timer Event yang ditambahkan pada Enemy?

.....
.....
.....



Timedoctor
Coding Academy

11

CORONA BUSTER: LET'S KILL THE VIRUS



Apa Yang Akan Kita Pelajari?

1. Menambahkan Laser
2. Menambahkan Shoot Button
3. Menambahkan Overlap Laser dan Virus



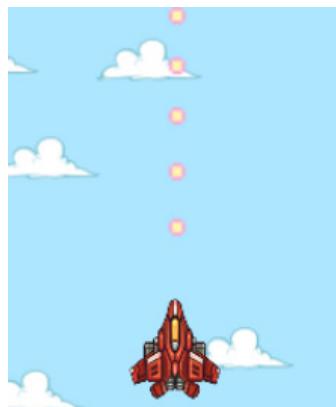
Pertemuan

11

Corona Buster

Pada pertemuan sebelumnya kita telah berhasil memunculkan Virus sebagai enemy. Sekarang kita perlu menambahkan senjata untuk membasi virus-virus itu.

Objek yang akan kita gunakan adalah Laser.
Selain itu, kita juga akan menambahkan Score untuk Player jika berhasil membunuh Virus itu.



Sebelum mulai membuat, kamu harus tahu **sifat** yang dimiliki Laser dan kode yang akan digunakan dulu ya! Jadi, laser akan memiliki sifat ini :

- 1) Laser bergerak mengikuti player -> **setPosition dengan parameter x dan y**
- 2) Laser akan dihapus setelah keluar layout -> **destroy**
- 3) Laser bergerak ke atas -> **setVelocityY bernilai negatif**
- 4) Laser muncul jika klik button Shoot -> **Conditional if**

Nah sekarang ayo kita mulai dengan membuat Kelas Laser terlebih dahulu!



Membuat Kelas Laser

Kelas Laser digunakan untuk menambahkan konfigurasi properti dan method dari Laser.

Kelas Laser akan memiliki method :

- **fire()** -> untuk memunculkan laser
- **erase()** -> untuk menghapus laser



Masih ingat cara membuat kelas baru?
Coba kamu buat kelas baru **di folder ui**, bernama **Laser.js**



Setelah membuat kelas Laser.js, Tulis kode ini ya!



Codes

```
import Phaser from 'phaser'

export default class Laser extends Phaser.Physics.Arcade.Sprite
{
    constructor(scene, x, y, texture) {
        super(scene, x, y, texture)
        this.setScale(2)
        this.speed = 200
    }

    fire(x, y)
    {
        this.setPosition(x, y - 50)
        this.setActive(true)
        this.setVisible(true)
    }
}
```



Kenapa method fire berisi parameter?

Kita menggunakan parameter x dan y karena **nilai x dan y Laser akan selalu berubah-ubah** tergantung pada posisi x dan y dari Player (karena player berpindah-pindah).



Kenapa pada setPosition, nilai y dikurangi 50?

Itu karena kita ingin **memunculkan laser tepat pada ujung pesawat.**

Coba kamu ubah kode y-50 menjadi y saja! Apa yang terjadi?

Nah sifat nomor 1) sudah terjawab. Ayo kita lanjutkan kodennya!



Pertemuan 11-Corona Buster: Let's Kill The Virus Laser Class

Setelah method fire(), kita juga akan menambahkan method erase() dan juga update().

Lanjutkan kode ini di dalam kelas Laser.js ya!



Codes

```
import Phaser from 'phaser'

export default class Laser extends Phaser.Physics.Arcade.Sprite
{
    //lanjutkan dari kode ini
    erase()
    {
        this.destroy()
    }

    update(time)
    {
        this.setVelocityY(this.speed * -1)
        if(this.y < -10) {
            this.erase()
        }
    }
}
```

Method erase() berisi pemanggilan method destroy yang berasal dari kelas Game Object pada Phaser.

Pada Scratch kode ini sama artinya dengan delete this clone.

destroy()

=

delete this clone



Lalu kapan method `erase()` itu dipanggil?
Perhatikan kondisi yang dibuat pada method `update()`!

```
if(this.y < -10) {  
    this.erase()  
}
```



Jadi Laser akan dihapus jika posisi `y` Laser kurang dari `-10`
Mengapa menggunakan `y < -10` ?

Nah sebenarnya kamu bisa mengubahnya dengan angka berapapun yang kurang dari angka 1, sehingga dia akan menghilang setelah keluar dari layout (`y = 1`)

Nah sekarang sifat nomor 2) sudah terjawab.
Sekarang ayo coba challenge ini agar lebih paham ya!



Challenge!

Coba ubah kode `y < -10` menjadi `y < 100` !



Bagaimana membuat laser bergerak ke atas?
Caranya adalah dengan mengatur `setVelocityY` menjadi negatif. perhatikan kode ini

```
this.setVelocityY(this.speed * -1)
```

Karena nantinya nilai speed akan diisi dengan angka positif, maka nilai `setVelocityY` akan menjadi negatif sehingga Laser bergerak ke atas

Bandingkan Virus dengan Laser



`setVelocityY(this.speed)`
bergerak ke bawah



`setVelocityY(this.speed * -1)`
bergerak ke atas



Pertemuan 11-Corona Buster: Let's Kill The Virus Add Laser

Itu sudah menjawab sifat nomor 3) kan?

Sekarang ayo kita munculkan laser ! Kembali ke **CoronaBusterScene**.



Menembakkan Laser

Cara untuk menembakkan laser adalah dengan menambahkan kondisi saat button shoot ditekan.

Tapi sebelum itu, coba kamu :**1) import kelas Laser** sendiri ya!
Setelah itu :**2) inisialisasi** properti baru untuk laser!



Codes

```
init()
{
    //tambahkan kode ini
    this.lasers = undefined
    this.lastFired = 0
}
```

Selain variabel lasers, kita juga menginisialisasi properti **lastFired bertipe data number untuk menyimpan jarak setiap laser yang ditembakkan**.

Selanjutnya : **3) load sprite sheet Laser** pada method preload()



Codes

```
this.load.spritesheet('laser', 'images/laser-bolts.png',
    {frameWidth: 16, frameHeight: 32,
     startFrame: 16, endFrame: 32})
)
```

Laser adalah object group yang dapat bergerak (dinamis), jadi kita akan menambahkannya di method create() menggunakan Physic Group.



Selanjutnya : **4) tambahkan Laser** pada method create()



Codes

```
this.lasers = this.physics.add.group({
    classType : Laser,
    maxSize : 10,
    runChildUpdate: true
})
```



Lalu dimana kita menambahkan kondisi untuk button shoot?

Lihat method movePlayer() !

5) Tambahkan parameter time dan kondisi baru pada method movePlayer() untuk button shoot !



Codes

```
movePlayer(player, time)
{ //tambahkan kode di bawah ini
    if ((this.shoot) && time > this.lastFired){
        const laser = this.lasers.get(0, 0, 'laser')
        if(laser){
            laser.fire(this.player.x, this.player.y)
            this.lastFired = time + 150
        }
    }
}
```



Coba reload game mu dan klik pada shoot button!

Apakah Laser muncul? Tidak kan?

Itu karena kita belum menambahkan **parameter time** saat pemanggilan method movePlayer() !

Sekarang coba kamu lihat **kembali ke pemanggilan method movePlayer() di method update()** !

6) Tambahkan parameter time di dalam method menjadi seperti ini : this.movePlayer(this.player, time)



Pertemuan 11-Corona Buster: Let's Kill The Virus Laser Overlap Enemy

Oke pasti sekarang laser sudah bisa dimunculkan.
Itu artinya sifat laser nomor 4) juga sudah terjawab ya !



Challenge!

Coba ubah kode `this.lastFired = time + 150` menjadi

1. `this.lastFired = time + 30`
2. `this.lastFired = time + 500`

Tapi...

Laser itu belum bisa membunuh virus-virus yang datang,
jadi apa yang harus dilakukan?



Ingatkah kamu bagaimana membuat 2 object bertabrakan
lalu menjalankan suatu method?

Ya kita akan menggunakan konsep overlaps!



Laser Overlaps Enemy



Yuk kita ingat-ingat dulu bagaimana membuat overlap antar 2 objek!



Warm Up!

Overlap berisi parameter :

```
Object1, //objek yg bersentuhan  
Object2, //objek yg disentuh  
collideCallback, //method yg dijalankan  
processCallback, //true atau false  
callbackContext //konteks terjadinya callback
```



Nah sebelum membuat overlap, siapkan method hitEnemy() dulu yaa. Buat kode ini paling bawah

Codes

```
hitEnemy(laser, enemy)
{
    laser.erase() //destroy laser yg bersentuhan
    enemy.die() //destroy enemy yg bersentuhan
}
```

Kode itu akan dipanggil pada overlaps agar dijalankan setelah Enemy dan Laser bersentuhan.



Do it Your Self !

Sekarang coba buat kode untuk overlap Enemy dan Laser sendiri ya!

Buatlah di method create() dan perhatikan parameter apa saja yang diperlukan pada overlap!



Challenge!

Ingin tidak cara menggerakkan dengan keyboard (cursor) ?
Coba kamu buat agar game bisa menembakkan laser menggunakan keyboard spasi !

Hint: tambahkan pada kondisi if di method movePlayer() !



Hari/Tanggal :

1. Pada method fire() di kelas Laser, kenapa setPosition berisi nilai y dikurangi 50?

.....
.....
.....

2. Apa fungsi kode ini ?

```
if(this.y < -10) {  
    this.erase()  
}
```

.....
.....
.....

3. Apa yang terjadi jika kode pada nomor 2 diganti menjadi `y < 100` ?

.....
.....
.....

4. Bagaimana kode untuk menambahkan laser pada method create() ?

.....
.....
.....

5. Method apa yang dipanggil saat Laser overlaps Enemy?

.....
.....
.....



Timedoctor
Coding Academy



Timedoctor
Coding Academy

12

CORONA BUSTER: Add Score Label



Apa Yang Akan Kita Pelajari?

1. Menambahkan Kelas Score Label
2. Memunculkan Score Label



Warm Up!

Sebelumnya kita telah membuat beberapa kelas, yaitu kelas Falling Object dan kelas Laser.

 **Masih ingatkah kamu apa fungsi dibuatnya Class ?
Class berfungsi sebagai template yang berisi properties
dan method yang bisa digunakan oleh banyak Object.**

Nanti kita juga akan membuat kelas untuk Score Label dan Life Label yang memiliki struktur method yang tidak jauh berbeda.

**Nah sekarang ayo kita mulai dengan menambahkan Score Label pada game,
Saat berhasil menembak virus maka score akan bertambah!**



Membuat Kelas Score Label

Score: 0

Coba sekarang kamu buat file baru di **folder ui, bernama ScoreLabel.js**

Kelas ScoreLabel akan memiliki method

- **setScore()** -> untuk mengatur nilai awal score dan teks
- **getScore()** -> untuk mengembalikan nilai score
- **add()** -> untuk menambahkan nilai score

Selanjutnya, tulis kode berikut ini di file ScoreLabel.js ya!



Codes

```
import Phaser from 'phaser'

const formatScore = (gameScore) => `Score: ${gameScore}`

export default class ScoreLabel extends Phaser.GameObjects.Text
{
    constructor(scene, x, y, skor, style)
    {
        super(scene, x, y, formatScore(skor), style)
        this.score = skor

    }

    setScore(skor)
    {
        this.score = skor
        this.setText(formatScore(this.score))
    }
}
```

Perhatikan kode ini yang ditulis setelah import!

```
const formatScore = (gameScore) => `Score: ${gameScore}`
```

Pada kode itu kita membuat sebuah **function** bernama **formatScore** yang berisi **parameter gameScore**.

Jika function ini dipanggil, maka akan mengembalikan (return) string dan nilai yang diberikan pada parameter gameScore (string interpolation).

Kode diatas adalah bentuk modern dari kode lama ini:

```
function formatScore(gameScore) {
    return `Score: ${gameScore}`
}
```



Pertemuan 12-Corona Buster: Add Score Label Score Label Class

Nah pada kelas constructor, kita membuat parameter `scene`, `x`, `y`, `skor`, dan `style`.

Tapi yang kita fokuskan hanya pada parameter `skor`.

Jika dilihat kode di dalam constructor ini:

```
super(scene, x, y, formatScore(skor), style)  
this.score = skor
```

Kita mengisi parameter `skor` menjadi -> `formatScore(skor)`

Dan menginisialisasi properti `score` (yaitu `this.score`) dengan nilai yang dimasukan ke parameter skor nantinya.

Artinya, parameter skor pada kelas ScoreLabel ini, menampung nilai skor yang akan selalu berubah sesuai dengan kondisi tertentu nantinya.

Nah sekarang lanjutkan kode ini di bawah `setScore(skor)`



Codes

```
getScore()  
{  
    return this.score  
}  
  
add(points)  
{  
    this.setScore(this.score + points)  
}
```

Coba kamu lihat method `add()` yang sudah dibuat :

```
add(points)  
{  
    this.setScore(this.score + points)
```



method `add()` berisi parameter `points` (`points`) yang akan ditambahkan dengan nilai yang ditampung di properti `score` (`this.score + points`)

Points menampung nilai skor yang ditambahkan saat Laser overlaps Enemy. Kita akan menggunakannya di CoronaBusterScene !



Menambahkan Score Label

Seperti biasa, kita harus **Import kelas ScoreLabel** pada CoronaBusterScene serta **membuat properti scoreLabel di method init()**.

1. Import kelas ScoreLabel.js

```
import ScoreLabel from '../ui/ScoreLabel.js'
```

2. Inisialisasi properti di method init()

```
this.scoreLabel = undefined
```

Nah selanjutnya kita perlu membuat method baru untuk membuat score label.



1. Method `createScoreLabel()`

Method **createScoreLabel()** digunakan **untuk menampilkan score label di layout**. Saat dipanggil nanti, terdapat parameter yang harus dimasukkan yaitu **x, y, dan score**.

Sekarang buatlah method baru (paling bawah) bernama **createScoreLabel()** seperti ini kode berikut ini.



Pertemuan 12-Corona Buster: Add Score Label Score Label Class

Codes

```
createScoreLabel(x, y, score)
{
    const style = { fontSize: '32px', fill: '#000' }
    const label = new ScoreLabel(this, x, y,
        score, style).setDepth(1)

    this.add.existing(label)

    return label
}
```

Terdapat 2 variabel lokal yang diinisialisasi pada kode diatas:

- **style** -> menyimpan gaya tulisan, kamu dapat mengganti font size dan warnanya sesuai keinginan!
- **label** -> memanggil kelas ScoreLabel dan mengisi parameternya
Jika dibandingkan antara parameter di kelas ScoreLabel.js dan ScoreLabel yang dipanggil diatas :

Kelas ScoreLabel.js

```
ScoreLabel(scene, x, y, skor, style)
```

ScoreLabel yang dipanggil di method createScoreLabel()

```
ScoreLabel(this, x, y, score, style)
```

Kita mengisi parameter :

- scene** -> dengan this (yaitu CoronaBusterScene)
- x** -> dengan x (nilainya diinputkan saat pemanggilan method)
- y** -> dengan y (nilainya diinputkan saat pemanggilan method)
- skor** -> dengan score (nilainya diinputkan saat pemanggilan method)
- style** -> dengan style (yang sudah diinisialisasi di atasnya)



Terdapat juga kode `setDepth(1)`
Masih ingat apa kegunaannya?

Setelah membuat method `createScoreLabel()`, selanjutnya panggil method itu pada method `create()`



Codes

```
create()
{
    //tambahkan kode ini
    this.scoreLabel = this.createScoreLabel(16, 16, 0)
}
```

Sebelumnya kita telah menginisialisasi variabel `scoreLabel` dengan nilai `undefined`.

Selanjutnya pada method `create()`, kita menginisialisasi `scoreLabel` dengan memanggil method `createScoreLabel()`. Perhatikan kode ini !

```
this.scoreLabel = this.createScoreLabel(16, 16, 0)
```

Inilah tadi yang disebutkan diatas, bahwa nilai x, y, dan score akan diinputkan saat pemanggilan method.

x y score

Score diisi dengan nilai 0 agar setiap permainan dimulai, score bernilai 0

Jika di game `BunnyJump`, itu sama dengan kode :

```
this.carrotsCollected = 0
```

Nah sekarang game mu sudah dapat menembakkan laser ke virus dan menambahkan score !



Tapi, apakah score sudah bertambah saat berhasil menembak virus?



2. Menambahkan Score

Game mu sudah bisa menembakkan laser, membunuh virus, dan menampilkan score label. Tapi kita harus menambahkan sedikit kode lagi agar saat virus overlap dengan laser, score bertambah 10.

Tambahkan kode ini pada method hitEnemy()



Codes

```
hitEnemy(laser, enemy)
{
    laser.erase()
    enemy.die()

    //tambahkan kode ini
    this.scoreLabel.add(10)
    if (this.scoreLabel.getScore() % 100 == 0){
        this.enemySpeed += 30
    }
}
```

Kode ini memanggil method add() dari kelas ScoreLabel

```
    this.scoreLabel.add(10)
```

Artinya setiap terjadi overlap antara laser dan enemy, maka score ditambah 10.

Lalu kondisi if ini digunakan untuk menambahkan kecepatan enemy setiap score bernilai kelipatan 100 (100, 200, 300, dst)

```
if (this.scoreLabel.getScore() % 100 == 0){
    this.enemySpeed += 30
}
```



Let's Explore

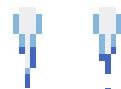
Ayo kita coba gunakan frame lain dari laser dengan mengatur kode load pada laser.

Spritesheet Laser memiliki width: 32 dan height: 32

Setiap frame pada laser (4 Frames) masing-masing memiliki width: 16 dan height: 16



Untuk mengatur bagian frame yang ingin digunakan, tambahkan properti startFrame dan endFrame.



Contohnya:

```
this.load.spritesheet('laser', 'images/laser-bolts.png',
    {frameWidth: 16, frameHeight: 32, startFrame: 16,
    endFrame: 32})
```

Coba dulu kode diatas, lalu ubah nilai-nilai propertinya sesuai dengan jenis laser yang ingin kamu gunakan!



Pertemuan 12-Corona Buster: Add Score Label Score Label Class

Hari/Tanggal : _____

1. Kelas apa saja yang kamu buat pada meeting ini?

.....
.....

2. Apa yang akan ditampilkan jika function formatScore pada kelas ScoreLabel dipanggil ?

.....
.....
.....

3. Bagaimana kode di dalam method `add(points)` pada kelas ScoreLabel?

.....
.....
.....

4. Apa fungsi kode ini?

```
const label = new ScoreLabel(this, x, y,  
    score, style).setDepth(1)
```

.....
.....

5. Dimana kamu menambahkan kode untuk menambahkan score 10? Bagaimana kodennya?

.....
.....
.....



Timedoctor
Coding Academy



Timedoctor
Coding Academy

13

CORONA BUSTER: DECREASE LIFE



Apa Yang Akan Kita Pelajari?

1. Menambahkan Life Label
2. Mengurangi Nilai Life



Corona Buster



Warm Up!



Kelas apa saja yang ditambahkan pada pertemuan sebelumnya?



Apa saja kegunaanya?



Nah, Hari ini kita akan menambahkan Life untuk Player dan membuat life dapat berkurang jika Player menyentuh Enemy

Ayo kita mulai! Kamu akan melihat banyak petunjuk yang harus kamu lakukan sendiri agar kamu lebih mengingatnya!



Membuat Kelas LifeLabel

Sama seperti Score, kita akan membuat kelas baru untuk Life. Coba kamu buat sendiri ya !



Do it Your Self !

1. Pada folder UI, buat file baru bernama LifeLabel !
2. Tambahkan struktur file seperti ScoreLabel.js (constructor, setLife, getLife, add(points), subtract(value))
3. Tambahkan kode : `const formatLife = (gameLife) => `Life: ${gameLife}``



Dengan melihat petunjuk ini, coba kamu buat kode di dalam method yang telah kamu buat di LifeLabel.js ya!
Kamu bisa centang di kolom untuk petunjuk sudah dikerjakan

1. constructor()

Pada method constructor, tambahkan :

- parameter** -> scene, x, y, lifePlayer, style
- super** -> scene, x, y, formatLife(lifePlayer), style
- properti** -> this.life = lifePlayer

2. setLife()

Pada method setLife(), tambahkan :

- parameter** -> lifePlayer
- properti** -> this.life = lifePlayer
- panggil method** -> this.setText(formatLife(lifePlayer))

3. getLife()

Pada method getLife(), tambahkan :

- return** -> this.life

4. add()

Pada method add(), tambahkan :

- parameter** -> points
- panggil method** -> this.setLife(this.life + points)

5. subtract()

Pada method subtract(), tambahkan :

- parameter** -> value
- panggil method** -> this.setLife(this.life - value)



Pertemuan 13 - Decrease Life Life Label Class

Kelas LifeLabel sudah selesai dibuat, yuk sekarang kembalikan ke CoronaBusterScene !



Menambahkan LifeLabel

Pertama kita harus **Import kelas LifeLabel** pada CoronaBusterScene serta **membuat properti lifeLabel di method init()**.

1. Import kelas LifeLabel.js

```
import LifeLabel from '../ui/LifeLabel.js'
```

2. Inisialisasi properti di method init()

```
this.lifeLabel = undefined
```

Nah selanjutnya kita perlu membuat method baru untuk membuat life label. Caranya sama dengan createScoreLabel lo!



Method createLifeLabel()

Buat method baru paling bawah ya !

Codes

```
createLifeLabel(x, y, life)
{
    const style = { fontSize: '32px', fill: '#000' }
    const label = new LifeLabel(this, x, y,
        life, style).setDepth(1)

    this.add.existing(label)

    return label
}
```



Kode pada method `createLifeLabel()` mirip dengan kode pada method `createScoreLabel` bukan?
Ya ! Karena method itu memiliki fungsi yang sama, yaitu **untuk menampilkan teks di Layout Game**



Do it Your Self !

Nah sekarang coba kamu panggil method `createLifeLabel()` pada method `create()` dengan ketentuan :

`x` -> 16
`y` -> 43
`life` -> 3

Coba kamu lihat pada game mu, apakah teks Life sudah muncul? Berapa life yang dimiliki player sejak game dimulai?

Selanjutnya adalah membuat life player **bisa berkurang dan bertambah**. Tapi sebelum itu kita buat game over scene dulu



Menambahkan Game Over Scene

Game over scene akan muncul jika Life = 0



Masih ingat bagaimana membuat game over scene pada game Bunny Jump?
Nah sekarang coba kamu buat game over scene mu sendiri.
Jika ingung, kamu bisa membuka project Bunny Jump atau tanyakan pada gurumu ya !

GAME OVER

SCORE : 0

Replay



Pertemuan 13 – Decrease Life Game Over Scene

Di dalam kelas GameOverScene, kamu akan membutuhkan kode berikut. Ikuti petunjuk ini ya !



Do it Your Self !

Jika kamu punya gambar lain untuk background, game over, atau replay button, kamu bisa mengantinya ya!

Kamu bisa tandai yg sudah dikerjakan di kolom kotak!

- Struktur kelas yang sama dengan CoronaBusterScene
(import phaser, export default class, constructor, preload, create)
- Load gambar background, gameover, dan replay
- Create gambar:
 - background (x = 200, y = 320)
 - gameover (x = 200, y =200)
- Deklarasi variabel global : var replayButton
Lalu Inisialisasi di method create() dengan kode
this.add.image pada x = 200 dan y = 530
- Panggil method once() pada replayButton :
`this.replayButton.once('pointerup', () => { this.
scene.start('corona-buster-scene') }, this)`



Konfigurasi main.js

Seperti biasa, kita perlu mengatur konfigurasi scene pada file main.js



1) Tambahkan kode: `import GameOverScene from './scenes/GameOverScene'`

2) Tambahkan GameOverScene pada properti scene

Menambahkan Score di GameOverScene

Sekarang ayo kita tambahkan nilai Score di GameOverScene !
Caranya mudah lo, hanya dengan 2 langkah !

1. Buat method init() di bawah constructor()



Codes

```
init(data)
{
    this.score = data.score
}
```

2. Tambahkan teks score pada method create()



Codes

```
this.add.text(80, 300, 'SCORE:', { fontSize:
    '60px', fill: '#000' })
//menambahkan nilai score
this.add.text(300, 300, this.score, { fontSize:
    '60px', fill: '#000'})
```



Decrease Life

Decrease Life atau **mengurangi nilai life** dibuat dengan cara membuat method baru bernama decreaseLife().

Pertemuan 13 - Decrease Life

Decrease Life



Membuat Method decreaseLife()

Sebelum ke kode, pahami dulu algoritma dari decreaseLife() ini ya !

Apa yang terjadi saat method decreaseLife() dipanggil?

1. Destroy Enemy yang terkena laser

2. Kurangi nilai life

3. Ganti Costume/Tampilan Player

Ayo coba kita tuliskan ke dalam kode !

Buat method decreaseLife() di CoronaBusterScene dan ikuti kode ini



Codes

```
decreaseLife(player, enemy)
{
    enemy.die()
    this.lifeLabel.subtract(1)

    if (this.lifeLabel.getLife() == 2){
        player.setTint(0xff0000)
    } else if (this.lifeLabel.getLife() == 1){
        player.setTint(0xff0000).setAlpha(0.2)
    } else if (this.lifeLabel.getLife() == 0) {
        this.scene.start('game-over-scene',
            {score: this.scoreLabel.getScore()})
    }
}
```

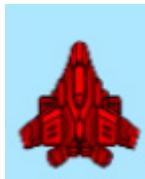


Kamu tahu tidak apa maksud kode yang ada di dalam kondisi if else?

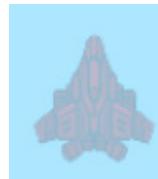
`setTint()` → perintah untuk mengubah warna objek

`0xff0000` → kode hexadecimal untuk warna merah

Jadi kondisi 1 dan kondisi 2 akan menghasilkan tampilan player seperti ini :



kondisi 1



kondisi 2



Bagaimana dengan kondisi 3 ?

```
else if (this.lifeLabel.getLife() == 0) {  
    this.scene.start('game-over-scene',  
        {score: this.scoreLabel.getScore()})  
}
```

Kondisi 3 membuat berpindah scene ke GameoverScene saat Life = 0,

Selain itu kita juga **memanggil nilai Score dengan method `getScore()`** karena nilai score itu akan ditampilkan di GameOverScene



Player Overlaps Enemy

Terakhir buat kode overlaps ini pada method create()



Codes

```
this.physics.add.overlap(this.player,  
    this.enemies, this.decreaseLife, null, this)
```



Pertemuan 13-Corona Buster: Decrease Life Score Label Class

Hari/Tanggal : _____

1. Method apa saja yang terdapat pada kelas LifeLabel?

.....
.....
.....

2. Apa fungsi kode ini pada GameOverScene?

```
this.add.text(300, 300, this.score, { fontSize:  
    '60px', fill: '#000'})
```

.....
.....

3. Apa fungsi kode ini pada method decreaseLife() ?

```
this.lifeLabel.subtract(1)
```

.....
.....

4. Apa kegunaan setTint() ?

.....
.....

5. Apa arti kode ini?

```
this.scene.start('game-over-scene',  
    {score: this.scoreLabel.getScore()})
```

.....
.....



Timedoctor
Coding Academy



Timedoctor
Coding Academy

14

CORONA BUSTER: INCREASE LIFE



Apa Yang Akan Kita Pelajari?

1. Menambahkan Handsanitizier
2. Menambahkan Nilai Life
3. Menambahkan Sound Effect



Corona Buster

Sedikit lagi game Corona Buster mu selesai!
Bagaimana perasaanmu setelah melewati perjalanan panjang
untuk menyelesaikan sebuah game?

**Jangan menyerah ya! Ayo kita
selesaikan game ini sedikit lagi!**



Nah, Hari ini kita akan **membuat Life Player bertambah saat mengambil Handsanitizer**
agar kamu bisa bertahan lebih lama di game dan game bisa jadi lebih seru !



Menambahkan Handsanitizer

Cara menambahkan handsanitizer sama dengan cara menambahkan enemy loh! Ayo ikuti langkah-langkah ini!

1. **Inisialisasi** properti Hansanitizer

```
this.handsanitizer = undefined
```

2. **Load** image Handsanitizer

```
this.load.image('handsanitizer', 'images/handsanitizer.png')
```

3. **Create** Handsanitizer

```
this.handsanitizer = this.physics.add.group ({  
    classType : Falling Object,  
    runChildUpdate : true  
})
```



1. Membuat Method spawnHandsantizer()



Masih ingat tidak bagaimana kode pada method spawnEnemy() ?

Kode pada spawnHandsantizer() juga hampir sama. Itu karena kita ingin memunculkan handsanitizer seperti memunculkan enemy

Sekarang coba kamu buat sendiri method spawnHandsantizer() ya !



Do it Your Self !

Kode pada spawnHandsantizer **mirip** dengan kode pada spawnEnemy, hanya saja ada **perbedaan** pada properti **speed** dan **rotation** yang bisa kamu isi dengan nilai sesukamu

```
const config = {  
    speed : 60,  
    rotation : 0  
}
```

Setelah membuat method spawnHandsantizer, sekarang kita panggil method itu pada method create() !



2. Panggil Method spawnHandsantizer()

Gunakan Timer Event juga untuk memanggil method spawnHandsantizer ya !

Tambahkan kode berikut ini pada method create()



Pertemuan 14 - Increase Life Spawn Handsanitizer



Codes

```
this.time.addEvent({
    delay: 10000,
    callback: this.spawnHandsantizer,
    callbackScope: this,
    loop: true
})
```

Sekarang coba kamu reload pada game mu dan pastikan Hansanitizer sudah muncul ya !

Selanjutnya kita akan membuat life player bertambah saat menyentuh Handsanitizer



Increase Life

Increase Life atau **menambahkan nilai life** dibuat dengan cara membuat method baru bernama **increaseLife()**.



Membuat Method increaseLife()

Method increaseLife() tidak jauh berbeda dengan method decreaseLife().

Jika pada method **decreaseLife()** kita menggunakan: **subtract**
Pada **increaseLife()** kita menggunakan: **add**

Buat method increaseLife() dan tambahkan kode berikut ini paling bawah ya !



Codes

```
increaseLife(player, handsanitizer)
{
    handsanitizer.die()
    this.lifeLabel.add(1)

    if (this.lifeLabel.getLife() >= 3){
        player.clearTint().setAlpha(2)
    }
}
```

Perhatikan kode di atas!

clearTint() → digunakan untuk menghapus warna sebelumnya dan mengembalikan warna objek seperti semula

Nah selanjutnya kita akan memanggil method increaseLife() pada kode Overlap !



3. Player Overlap Handsanitizer



Masih ingat tidak bagaimana kode ntuk membuat overlap objek?

Coba kamu buat sendiri ya !



Do it Your Self !

Buatlah kode overlap untuk Player dan Handsanitizer, lalu panggil method increaseLife() !

Kamu bisa melihat kode sebelumnya ya !

Game mu sudah selesai ! Ayo mainkan dan kumpulkan score sebanyak-banyaknya!



Menambahkan Sound Effect

Agar game mu lebih seru ayo kita tambahkan sound effect!

Kita akan beberapa sound effect yaitu **laser, destroy enemy, get handsanitizer, backsound, dan game over.**

1) Download semua sound pada link ini:

http://tiny.cc/meeting14_sfx dan letakkan pada folder sfx di folder public

2) Lalu load sound pada method preload()



Codes

```
this.load.audio('laserSound', 'sfx/sfx_laser.ogg')
this.load.audio('destroySound', 'sfx/destroy.mp3')
this.load.audio('handsanitizerSound', 'sfx/handsanitizer.mp3')
```



Laser Sound Effect

Sound effect dari Laser akan dimulai saat button Shoot ditekan. Jadi kode play sound akan dibuat di dalam kondisi if(laser).

Lihat method movePlayer() dan tambahkan kode berikut ini pada kondisi if(laser) seperti yang ditandai pada comment



Codes

```
if(laser) {
    laser.fire(this.player.x, this.player.y)
    this.lastFired = time + 150
    //tambahkan kode di bawah ini
    this.sound.play('laserSound')
```



Destroy Sound Effect

Saat Laser collides Enemy, maka play sound effect destroy. Caranya adalah dengan **menambahkan kode pada method hitEnemy() seperti yang ditunjukkan pada comment.**



Codes

```
hitEnemy() {  
    laser.erase()  
    enemy.die()  
    //tambahkan kode di bawah ini  
    this.sound.play('destroySound')
```



Do it Your Self !

Kita juga akan menambahkan sound effect handsanitizer saat Handsanitizer berhasil didapatkan.

Tambahkan kode pada method increaseLife()



Game Backsound

Ayo kita tambahkan backsound juga pada game Corona Buster !
Kamu bisa memilih Audio mana yang ingin kamu gunakan pada folder Backsound ya!

1) Inisialisasi properti di method init()



Codes

```
this.backsound = undefined
```

2) Load backsound yang ingin kamu gunakan pada method preload(). Misalnya menggunakan backsound SkyFire



Codes

```
this.load.audio('backsound', 'sfx/backsound/SkyFire.ogg')
```

3) Create backsound di method create dengan kode seperti ini



Codes

```
this.backsound = this.sound.add('backsound')
var soundConfig = {
    loop: true
}
this.backsound.play(soundConfig)
```

Nah Sekarang coba jalankan game mu dan pastikan backsound sudah dimainkan berulang-ulang



Game Over Sound Effect

Terakhir, ayo kita buat semua sound berhenti saat Player Game Over dan Play sound game over.

Load game over sound effect pada method preload()



Codes

```
this.load.audio('gameOverSound', 'sfx/gameover.ogg')
```

Lalu create sound pada method decreaseLife() pada kondisi game over seperti yang ditandai pada comment.



Codes

```
} else if (this.lifeLabel.getLife() == 0) {  
    this.scene.start('game-over-scene', {score:  
        this.scoreLabel.getScore() })  
    //tambahkan kode di bawah ini  
    this.sound.stopAll()  
    this.sound.play('gameOverSound')
```



Challenge!

Coba kamu ganti sound Game Over dengan audio yang lainnya dengan mencari sendiri di internet ya !



Pertemuan 14 - Increase Life Spawn Handsanitizer

Hari/Tanggal : _____

1. Method apa saja yang terdapat pada kelas LifeLabel?

.....
.....

2. Sebutkan langkah-langkah untuk menambahkan Handsanitizer !

.....
.....

3. Apa fungsi kode ini pada method increaseLife() ?

`this.lifeLabel.add(1)`
.....
.....

4. Apa fungsi method clear TInt() ?

.....
.....

5. Sound effect apa saja yang kamu tambahkan?

.....
.....



Timedoctor
Coding Academy



Timedoctor
Coding Academy

15

GHOSTBUSTER: INTERMEDIATE 2 TEST



Apa Yang Akan Kita Pelajari?

1. Test Intermediate 2
2. Soal akan diberikan oleh Teacher.



Timedoctor
Coding Academy

16

GHOSTBUSTER: INTERMEDIATE 2 TEST



Apa Yang Akan Kita Pelajari?

1. Test Intermediate 2
2. Soal akan diberikan oleh Teacher.



Timedoctor
Coding Academy

INTERMEDIATE 3





Timedoctor
Coding Academy

1

MATH FIGHTER: SETUP THE ANIMATION



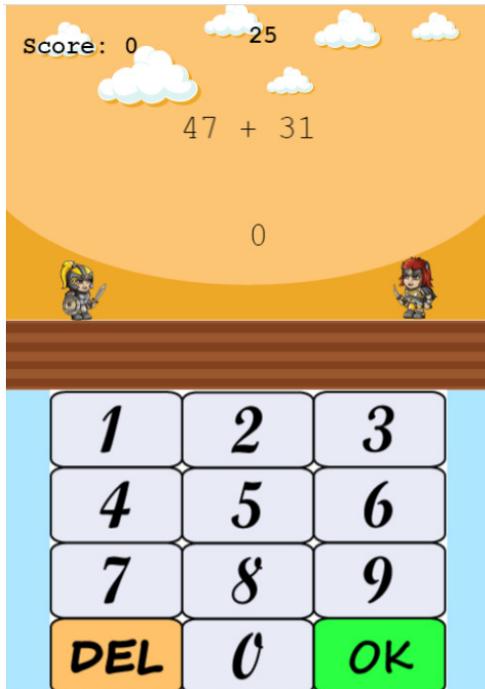
Apa Yang Akan Kita Pelajari?

1. Menambahkan Player, Enemy, dan Slash
2. Membuat Animasi Player dan Enemy



Math Fighter

MATH FIGHTER



Math Fighter adalah game matematika yang kemenangannya ditentukan oleh kemampuan berhitung mu! Jika jawabanmu benar, maka kamu bisa menyerang musuh. Tapi, jika kamu salah maka musuh lah yang akan menyerangmu!

Ayo ikuti petunjuk ini untuk membuat persiapan mu sendiri ya ! Jika bingung, kamu bisa lihat pada game sebelumnya.



Preparation

Seperti biasa kita harus melakukan beberapa persiapan dulu sebelum mulai coding!



1. Prepare Template

Kita sudah melakukan persiapan berulang kali, saat Intermediate 2, sekarang coba kamu lakukan sendiri ya !



Do it Your Self !

Kamu bisa menandai bagian yang sudah dikerjakan agar tidak bingung !

- Copy paste** Template Phaser dan **Rename** menjadi *Math Fighter*
- Download assets di : tiny.cc/mathfighter_assets
- Buat Folder **Public > Images > Masukkan Semua Assets**
- Pada **Index.html**, Ubah Judul Tab menjadi *Math Fighter*
- Pada **Folder scenes**, Buat file *MathFighterScene.js*
- Pada **main.js**, **Konfigurasi** *MathFighterScene.js*
- Import MathFighterScene** dan tambahkan pada **scene**
- Atur ukuran layout game, **width : 480** dan **height : 640**
- Tambahkan **properti Scale** pada main.js

```
scale: {  
    mode: Phaser.Scale.FIT,  
    autoCenter: Phaser.Scale.CENTER_BOTH  
},
```



2. Struktur Kelas MathFighterScene

Selanjutnya buatlah struktur kelas MathFighterScene.js



Codes

```
import Phaser from 'phaser'  
export default class MathFighterScene extends Phaser.Scene  
{  
    constructor()  
    {  
        super('math-fighter-scene')  
    }  
    init()  
    {  
    }  
    preload()  
    {  
    }  
    create()  
    {  
    }  
    update()  
    {  
    }  
}
```

Struktur kelas pada MathFighterScene.js sama seperti struktur kelas di game lainnya.

**Sekarang kamu akan membuat sendiri kode untuk load assets.
Jangan khawatir! Ikuti petunjuk ini ya!**



Kode Load Assets

Assets terdiri dari image dan sprite sheet. Ikuti tabel ini dan load assets satu per satu ya !

| IMAGE | KEY |
|------------------|--------------|
| bg_layer1.png | 'background' |
| fight-bg.png | 'fight-bg' |
| tile.png | 'tile' |
| start_button.png | 'start-btn' |

| SPRITE SHEET | KEY | FRAME HEIGHT | FRAME WIDTH |
|--------------|-----------|--------------|-------------|
| warrior1.png | 'player' | 80 | 80 |
| warrior2.png | 'enemy' | 80 | 80 |
| numbers.png | 'numbers' | 71.25 | 131 |
| slash.png | 'slash' | 88 | 42 |

Ingat!

Sprite sheet **memiliki frameWidth dan frameHeight yang sama** untuk setiap frame di dalamnya.

Jadi untuk mencari nilai width dan height dari 1 frame kamu harus **membagi total width dengan jumlah frame**



Bagaimana cara menghitungnya?



Pertemuan 1 - Setup The Animation

Load Assets

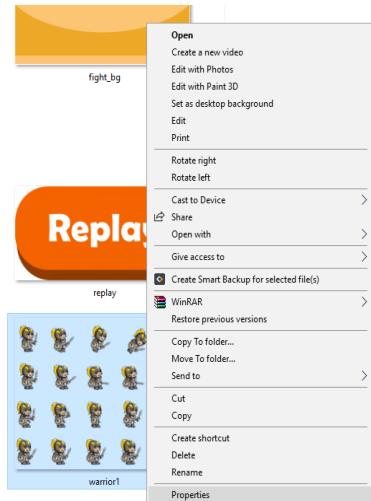
Cara mencari frameWidth dan frameHeight

1. Cari file image melalui File Explorer

> This PC > Desktop > Intermediate2 > Meeting12 > MathFighter > public > images

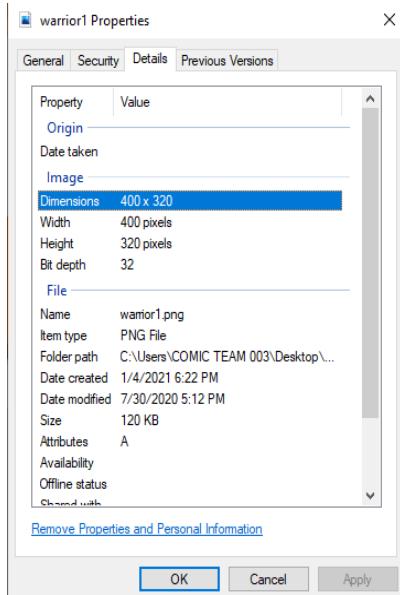


2. Klik kanan pada image > pilih Properties



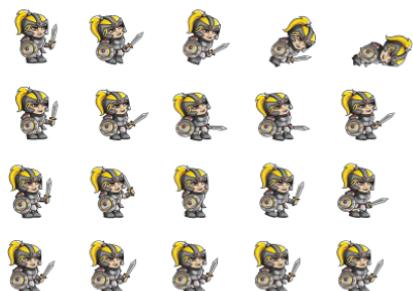


3. Pada Properties, pilih tab Details dan **lihat ukuran width dan height** dari gambar tersebut.



Pada sprite sheet **warrior1.png**,
ukuran **width** = 400 pixels
ukuran **height** = 320 pixels
jumlah **frame horizontal** = 5
jumlah **frame vertical** = 4

Jadi,
frameWidth → $400 : 5 = 80$
frameHeight → $320 : 4 = 80$



Itulah yang membuat ukuran **frameWidth** dan **frameHeight** pada saat load sprite sheet adalah 80



Pertemuan 1 - Setup The Animation

Create Image and Sprite



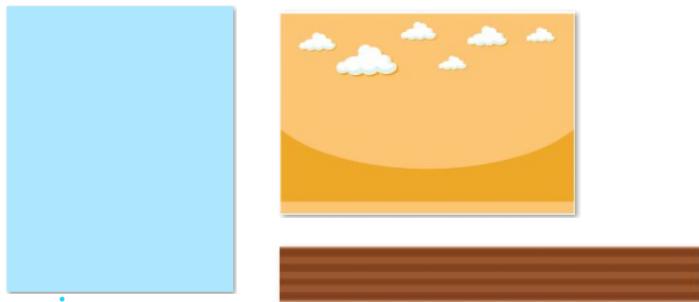
Create Image and Sprite

Setiap assets akan dimunculkan dengan cara yang berbeda, kita mulai dari yang paling mudah dulu ya !



1. Create Background, Fight-Bg, and Tile

Pertama kita akan menambahkan background, fight-bg, dan tile. Tile akan menjadi objek Physics yang diam (Static) karena akan menjadi pijakan dari Player dan Enemy.

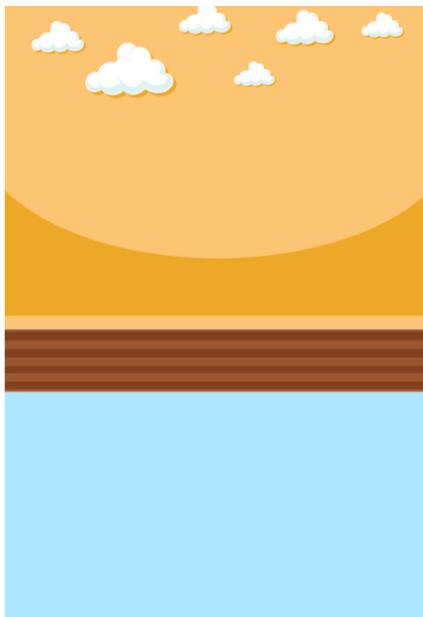


Codes

```
create()
{
    //create background
    this.add.image(240, 320, 'background')
    //create fight-bg
    const fight_bg = this.add.image(240, 160, 'fight-bg')
    //create tile
    const tile = this.physics.add.staticImage(240,
        fight_bg.height - 40, 'tile')
}
```



Coba jalankan projectmu dengan **npm run start pada command line** dan pastikan hasilnya seperti ini !



Selanjutnya kita akan menambahkan Player, Enemy, Slash, dan Numbers.

Tapi sebelum itu kita perlu membuat properti gameHalfWidth dan gameHalfHeight **untuk mencari titik tengah layout** sehingga memudahkan penempatan objek lainnya.

Inisialisasi properti berikut ini pada method init()



Codes

```
init()
{
    this.gameHalfWidth = this.scale.width * 0.5
    this.gameHalfHeight = this.scale.height * 0.5
}
```



Pertemuan 1 - Setup The Animation

Create Image and Sprite

2. Create Player

Kita akan menambahkan player menggunakan kelas Sprite dari Arcade Physics Body.

Pertama, inisialisasi dulu properti player pada method init()

Buat kode ini pada method init()



Codes

```
this.player = undefined
```

Lalu tambahkan sprite player pada method create()

Buat kode ini pada method create()



Codes

```
this.player = this.physics.add.sprite(  
    this.gameHalfWidth - 150,  
    this.gameHalfHeight - 200, 'player')
```

Selanjutnya kita akan membuat Player bisa berpijak diatas Tile.

1. Menambahkan Collider

Buat kode ini pada method create()



Codes

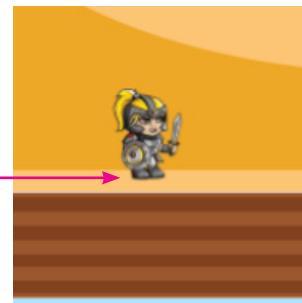
```
this.physics.add.collider(this.player, tile)
```



Coba kamu reload dan lihat apakah Player sudah muncul dan berpijak di atas Tile dengan benar?



Apakah hasilnya seperti gambar di samping?
Kira-kira apa penyebabnya?



80



Hal itu terjadi karena kita menggunakan **frameHeight**:
80 saat load sprite sheet, yang melebihi batas kaki Player.

80



Lalu bagaimana mengatasinya?

Caranya dengan menambahkan method **setOffset** saat create Player.



Vocabulary

setOffset(x,y) digunakan untuk menyesuaikan posisi objek dengan cara menambah atau mengurangi x dan y.

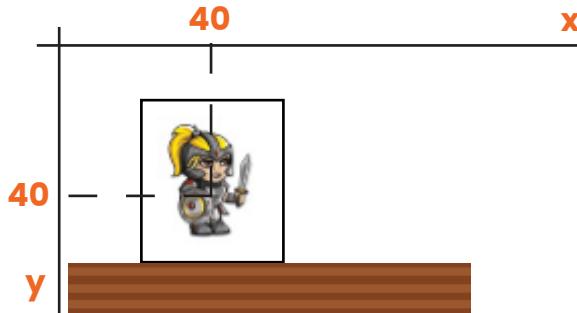
Perubahan yang dibuat oleh setOffset akan terlihat saat objek mengalami collides.



Pertemuan 1 - Setup The Animation

Create Image and Sprite

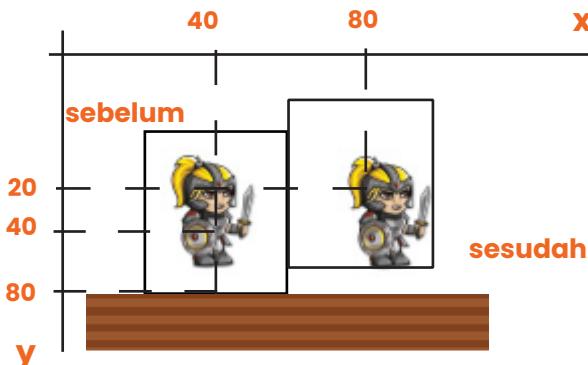
Perhatikan gambar berikut ini!



Misalnya posisi game object saat di create ada di :
x = 40 dan y = 40 dan terlihat seperti gambar diatas.

Lalu kita **menambahkan method setOffset(40,-20)**,
Maka sisi objek yang dapat bersentuhan berubah menjadi :
offset x = **40** → artinya **ditambah 40 px (semakin ke kanan)**
offset y = **-20** → artinya **dikurangi 20 px (semakin ke atas)**

Jika dibuktikan, perubahan yang dialami bisa digambarkan menjadi seperti ini.



Tanda kotak → Object Body (muncul sesuai offset)
Gambar Hero → Game Object (tetap muncul pada posisi awal)



Karena body object dipindah ke atas, maka object akan tetap jatuh sampai kaki Hero menyentuh Platform.

2. Menambahkan setOffset dan setBounce

Setelah melihat penjelasan tentang setOffset , ayo kita coba gunakan pada kode menambahkan player !

Tambahkan setOffset di method create() seperti yang ditunjukkan pada comment



Codes

```
this.player = this.physics.add.sprite(  
    this.gameHalfWidth - 150,  
    this.gameHalfHeight - 200, 'player')  
    //tambahkan kode ini  
    .setOffset(-50, -8)
```

Sekarang coba jalankan game mu dan lihat perbedaannya!

Selain itu kita akan menambahkan efek bounce (memantul) saat Player jatuh di atas Tile.

Tambahkan setBounce seperti yang ditunjukkan pada comment



Codes

```
this.player = this.physics.add.sprite(  
    this.gameHalfWidth - 150,  
    this.gameHalfHeight - 200, 'player')  
    .setOffset(-50, -8)  
    //tambahkan kode ini  
    .setBounce(0.2)
```

Kamu juga bisa mengatur nilai setBounce sesukamu ya!



Pertemuan 1 - Setup The Animation

Create Image and Sprite



3. Create Enemy

Cara menambahkan Enemy juga sama dengan menambahkan Player. Hanya saja arah hadap Enemy berbeda dengan Player yaitu menghadap ke kiri.

Sekarang coba kamu tambahkan Enemy mu sendiri ya ! Ikuti petunjuk di bawah ini agar lebih mudah menyelesaiannya.



Do it Your Self !

Kamu bisa menandai bagian yang sudah dikerjakan agar tidak bingung !

Inisialisasi properti enemy dengan undefined

```
this.enemy = undefined
```

Tambahkan Enemy pada method create() di titik :

```
x = this.gameHalfWidth + 150
```

```
y = this.gameHalfHeight - 200
```

Tambahkan setOffset(50,-8), setBounce(0.2), dan setFlipX(true)

Tambahkan Collider Enemy dengan Tile

Coba jalankan game mu dan pastikan hasilnya seperti ini ya!





4. Create Slash

Sekarang kita juga menambahkan slash, tapi setelah ini kita belum dapat menggunakannya ya karna kita akan menggunakannya di pertemuan berikutnya.

Inisialisasi properti slash pada method init()



Codes

```
this.slash = undefined
```

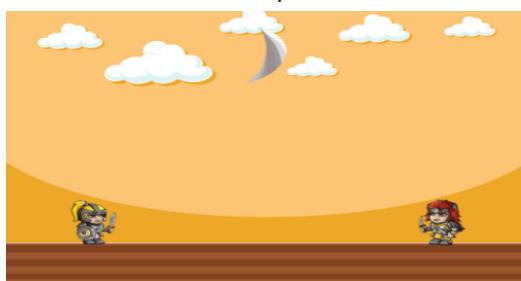
Selanjutnya buat kode ini pada method create()



Codes

```
this.slash = this.physics.add.sprite(240, 60, 'slash')  
    . setActive(false)  
    . setVisible(false)  
    . setGravityY(-500)  
    . setOffset(0, -10)  
    . setDepth(1)  
    . setCollideWorldBounds(true)
```

Jika kamu ubah setVisible() menjadi true. Maka kamu akan melihat Slash berada di atas seperti ini





Pertemuan 1 - Setup The Animation

Create Image and Sprite

Sekarang kita akan menambahkan animasi!



Create Animation

Saat game berjalan, pasti akan lebih menarik jika Player dan Enemy memiliki animasi bukan?

Nah sekarang kita akan **membuat animasi saat Player dan Enemy Standby, Attack, Hit, dan Die** dengan memanfaatkan frame-frame yang ada



1. Animasi Player

Animasi Die



Frame: 0 1 2 3 4

Animasi Hit



Frame: 5 6 7 8 9

Animasi Attack



Frame: 10 11 12 13 14



Animasi Standby



Frame: 15 16 17 18 19

Setelah membagi frame-frame Player sesuai animasi nya, ayo sekarang kita buat kode untuk animasi Player !

Buat method baru bernama `createAnimation()` dan tulis kode berikut ini



Codes

```
createAnimation(){
    this.anims.create({
        key: 'player-die',
        frames: this.anims.generateFrameNumbers('player',
            { start: 0, end: 4 }),
        frameRate: 10,
    })
    this.anims.create({
        key: 'player-hit',
        frames: this.anims.generateFrameNumbers('player',
            { start: 5, end: 9 }),
        frameRate: 10,
    })

    this.anims.create({
        key: 'player-attack',
        frames: this.anims.generateFrameNumbers('player',
            { start: 10, end: 14 }),
        frameRate: 10
    })
}
```



Pertemuan 1 - Setup The Animation

Create Image and Sprite

Codes

```
//lanjutan  
this.anims.create({  
    key: 'player-standby',  
    frames: this.anims.generateFrameNumbers('player',  
        { start: 15, end: 19 }),  
    frameRate: 10,  
    repeat: -1  
})
```



2. Animasi Enemy

Enemy juga memiliki jumlah dan struktur frame yang sama. Jadi kamu bisa melihat kode nya pada animasi Player.

Tambahkan kode animasi Enemy di dalam method `createAnimation()` setelah kode animasi Player.



Do it Your Self !

Coba kamu buat sendiri ya! Kamu hanya perlu mengganti beberapa kode seperti :

key -> 'enemy-die', 'enemy-hit', 'enemy-attack',
'enemy-standby'

key pada frames -> 'player' diganti menjadi 'enemy'



Selanjutnya kita akan panggil method `createAnimation()` pada method `create()`



Codes

```
this.createAnimation()
```

Tapi, belum terjadi perubahan apapun pada Player dan Enemy karena kita **belum menggunakan animasi tersebut pada suatu kondisi.**

Kita akan lanjutkan pada pertemuan berikutnya!



Pertemuan 1 - Setup The Animation Spawn Handsanitizer

Hari/Tanggal : _____

1. Image dan Sprite sheet apa saja yang ditambahkan pada meeting ini?

.....
.....
.....

2. Apa fungsi kode ini pada method init() ?

```
this.gameHalfWidth = this.scale.width * 0.5  
this.gameHalfHeight = this.scale.height * 0.5
```

.....
.....
.....

3. Bagaimana urutan langkah untuk menambahkan Player dan Enemy?

.....
.....

4. Apa fungsi method setOffset() ?

.....
.....

5. Animasi apa saja yang dimiliki Player dan Enemy?

.....
.....
.....



Timedoctor
Coding Academy



Timedoctor
Coding Academy

2

MATH FIGHTER: START GAME AND CREATE BUTTON



Apa Yang Akan Kita Pelajari?

1. Menambahkan Button Game Start
2. Menambahkan Button Number



Math Fighter



Warm Up!



Apa saja yang sudah kamu tambahkan pada pertemuan sebelumnya?

Apakah slash sudah bisa digunakan?

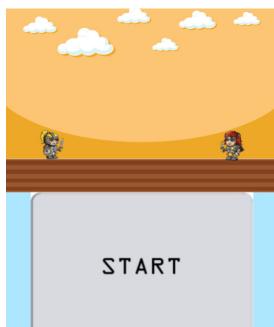
Apakah Player dan Enemy sudah beranimasi?



Lalu apa yang akan ditambahkan kali ini?

Pada pertemuan kali ini kita akan menambahkan Button Start dan Button Number

Math Fighter Game akan memiliki **Timer**. Maka dari itu kita akan memerlukan button Start agar Timer mulai menghitung mundur setelah button Start di klik.



Kebanyakan Game terutama Timed Game (Game dengan Timer) pasti memiliki Button Start agar Player bisa bersiap terlebih dahulu sebelum Timer mulai menghitung.



Ayo sekarang kita buat game Math Fighter memiliki Button Start!



Create Start Game

Ayo kita buat start game! Kita akan menambahkan start button yang harus di klik untuk memulai game.

Pastikan dulu kamu sudah load image start button pada method preload() ya !



Codes

```
this.load.image('start-btn', 'images/start_button.png')
```

Nah selanjutnya,, Inisialisasi properti startGame di method init()



Codes

```
this.startGame = false  
this.questionText = undefined  
this.resultText = undefined
```

Kita memberi nilai awal startGame berupa tipe data Boolean. **startGame dinonaktifkan terlebih dahulu ketika game dijalankan,** Nanti kita akan **mengubahnya menjadi true ketika button start di klik**

Pada kode di atas kita juga menambahkan properti untuk menyimpan teks pertanyaan (questionText) dan jawaban (resultText)



Pertemuan 2 - Start Game and Create Button

Create Start Game

Selanjutnya, Buat method baru bernama gameStart()



Codes

```
gameStart()
{
    this.startGame = true
    this.player.anims.play('player-standby', true)
    this.enemy.anims.play('enemy-standby', true)

    this.resultText = this.add.text(this.gameHalfWidth,
        400, '0', { fontSize: '32px', fill: '#000' })
    this.questionText = this.add.text(this.gameHalfWidth,
        200, '0', { fontSize: '32px', fill: '#000' })
}
```

Setelah memberi nilai kembali pada startGame menjadi true, kita juga mengatur player dan enemy agar beranimasi standby dan menambahkan teks resultText dan questionText.

Selanjutnya tambahkan button start pada method create() dan kita akan memanggil method gameStart() juga disini.



Codes

```
let start_button = this.add.image(this.gameHalfWidth,
this.gameHalfHeight + 181, 'start-btn').setInteractive()

start_button.on('pointerdown', () => {
    this.gameStart()
    start_button.destroy()
}, this)
```



Pada kode ini :

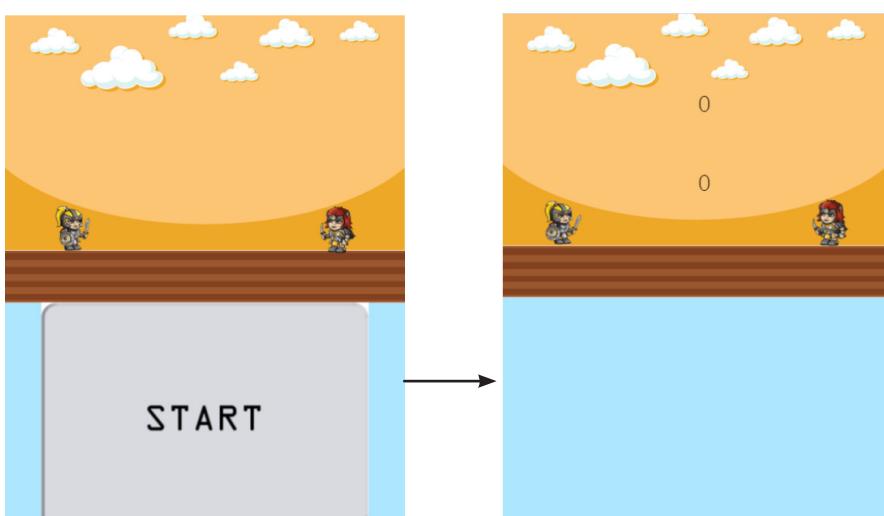
```
start_button.on('pointerdown', () => {
    this.gameStart()
    start_button.destroy()
}, this)
```

Kita mengatur agar **“Ketika button start di klik, panggil method gameStart() dan destroy start button”**



Apakah kondisi di atas sudah berhasil dijalankan pada game mu?

Apakah Player dan Enemy sudah beranimasi stand-by?



Jalankan game mu, lakukan debugging, dan pastikan semua kode sudah berjalan dengan benar ya!

Selanjutnya kita akan menambahkan Button Number



Create Button Number



Create Button Number

Kita akan memerlukan method baru untuk Create Buttons dan akan dipanggil setelah Button Start di klik

Tapi sebelum itu, inisialisasi properti button terlebih dulu ya!



Codes

```
init()
{
    //lanjutkan kode pada method init
    this.button1 = undefined
    this.button2 = undefined
    this.button3 = undefined
    this.button4 = undefined
    this.button5 = undefined
    this.button6 = undefined
    this.button7 = undefined
    this.button8 = undefined
    this.button9 = undefined
    this.button0 = undefined
    this.buttonDel = undefined
    this.buttonOk = undefined
}
```

Sekarang kita menambahkan frame button numbers satu per satu.



Mengapa tidak menambahkan sprite sheetnya sekaligus?
Jika ditambahkan sekaligus, maka kita tidak bisa klik salah satu button saja.





Bagaimana cara menambahkan setiap button agar berada di tengah-tengah layout dan setiap frame tersusun rapi?

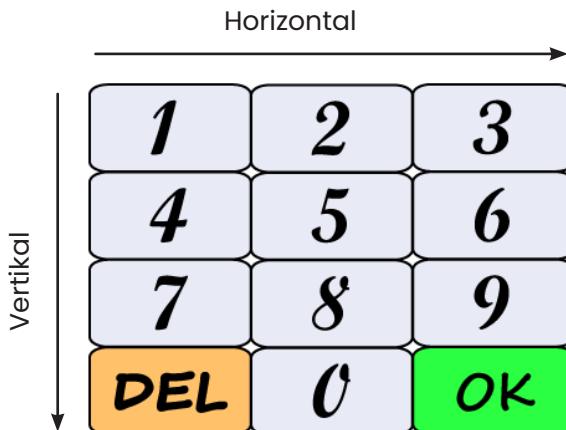
Coba kamu lihat strategi ini ya!

Masalah : Meletakkan numbers di posisi tengah-tengah layout

Perhatikan sprite sheet numbers!

Secara horizontal -> ada 3 Frames

Secara vertikal -> ada 4 Frames



Jika kita ingin meletakkan semua numbers di tengah dan menghasilkan posisi seperti pada sprite sheet, akan lebih mudah jika kita memilih deretan numbers yang berjumlah ganjil, yaitu deretan secara horizontal.

Seperti 3 anak yang sedang berbaris, agar barisan rapi maka anak yang di tengah harus mengukur jarak dengan tangannya sendiri





Pertemuan 2 – Start Game and Create Button Create Button Number

Solusi :

- Membuat Kelompok Tengah Terlebih Dahulu

Frame yang dibuat pertama adalah frame yang berada di tengah, yaitu **frame 1 (button2), frame 4 (button5), dan frame 7 (button8)**

- Berikan jarak

Saat ingin membuat button lainnya (Kelompok Kiri dan Kelompok Kanan) **buat pada posisi x dengan patokan Kelompok Tengah**



Sudah terbayang?

Ayo kita coba langsung ya! Kita akan memanfaatkan properti gameHalfWidth yang telah dibuat sebelumnya!

Buat method baru bernama **createButtons()** dan tulis kode ini



Codes

```
createButtons() {  
    const startPositionY = this.scale.height - 246  
    const widthDifference = 131  
    const heightDifference = 71.25  
  
    this.button2 = this.add.image(this.gameHalfWidth,  
        startPositionY, 'numbers', 1)  
        .setInteractive().setData('value', 2)  
    this.button5 = this.add.image(this.gameHalfWidth,  
        this.button2.y + heightDifference, 'numbers', 4)  
        .setInteractive().setData('value', 5)  
    this.button8 = this.add.image(this.gameHalfWidth,  
        this.button5.y + heightDifference, 'numbers', 7)  
        .setInteractive().setData('value', 8)  
    this.button0 = this.add.image(this.gameHalfWidth,  
        this.button8.y + heightDifference, 'numbers', 10)  
        .setInteractive().setData('value', 0)  
}
```



Panggil method `this.createButton()` pada method `gameStart()`



Codes

```
this.createButtons()
```

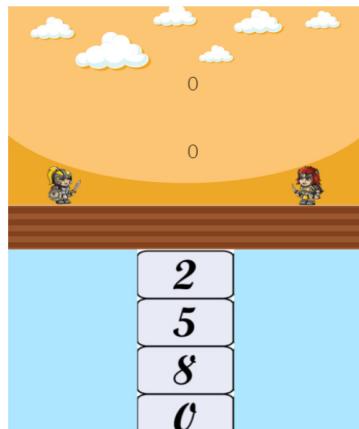
Jika dijalankan maka hasilnya seperti gambar di samping.

Kalian berhasil membuat **Kelompok Tengah numbers!**



Tapi apakah kamu paham bagaimana posisi numbers itu tepat dan rapi?

Itu karena kita menggunakan bantuan variabel !

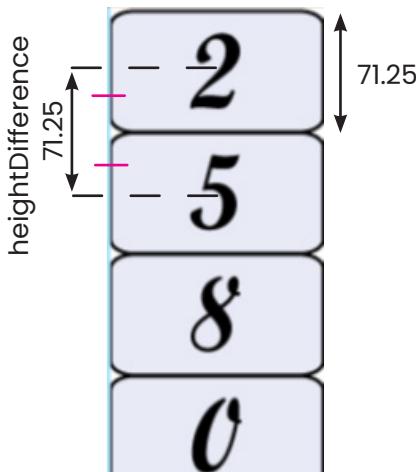


Variabel startPositionY dan **heightDifference** digunakan untuk menempatkan numbers di posisi **vertical** (ke bawah)



Kenapa `heightDifference` bernilai 71.25?

Coba kamu ingat height dari setiap frame bernilai 71.25 juga bukan?

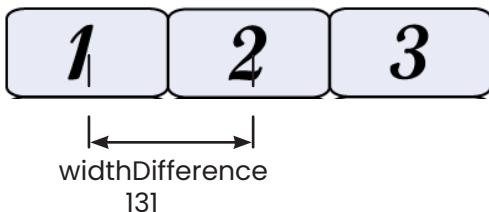


Karena kita akan meletakkan setiap numbers berdempetan, jadi kita hanya perlu menentukan jarak antara posisi Y button2 dengan posisi Y button5 menggunakan nilai yang sama dengan tinggi (height) frame

Create Button Number



Sedangkan widthDifference digunakan untuk menempatkan numbers di posisi **horizontal** (ke samping)



Sekarang kita akan menggunakan widthDifference untuk membuat button lainnya.

Lanjutkan kode tadi dengan kode berikut ini ya!



Codes

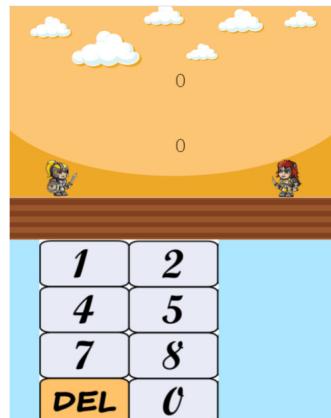
```
createButtons() {
    //lanjutan
    this.button1 = this.add.image(this.button2.x -
        widthDifference, startPositionY, 'numbers', 0)
        .setInteractive().setData('value', 1)
    this.button4 = this.add.image(this.button5.x -
        widthDifference, this.button1.y +
        heightDifference, 'numbers', 3)
        .setInteractive().setData('value', 4)
    this.button7 = this.add.image(this.button8.x -
        widthDifference, this.button4.y +
        heightDifference, 'numbers', 6)
        .setInteractive().setData('value', 7)
    this.buttonDel = this.add.image(this.button0.x -
        widthDifference, this.button7.y +
        heightDifference, 'numbers', 9)
        .setInteractive().setData('value', 'del')
}
```



Jika dijalankan maka hasilnya seperti gambar di samping.
Sekarang kalian berhasil membuat **Kelompok Kiri numbers!**

Sedikit lagi kamu berhasil membuat semua numbers loh!

Ayo kita buat kelompok yang terakhir yaitu **Kelompok Kanan!**



Lanjutkan kode tadi dengan kode berikut ini ya!



Codes

```
createButtons() {  
    //lanjutan  
    this.button3 = this.add.image(this.button2.x +  
        widthDifference, startPositionY, 'numbers', 2)  
        .setInteractive().setData('value', 3)  
    this.button6 = this.add.image(this.button5.x +  
        widthDifference, this.button3.y +  
        heightDifference, 'numbers', 5)  
        .setInteractive().setData('value', 6)  
    this.button9 = this.add.image(this.button8.x +  
        widthDifference, this.button6.y +  
        heightDifference, 'numbers', 8)  
        .setInteractive().setData('value', 9)  
    this.buttonOk = this.add.image(this.button0.x +  
        widthDifference, this.button9.y +  
        heightDifference, 'numbers', 11)  
        .setInteractive().setData('value', 'ok')  
}
```



Pertemuan 2 – Start Game and Create Button

Create Button Number

Hasilnya akan menjadi seperti ini



Kalian sudah berhasil menambahkan button, tapi itu hanyalah gambar yang bisa di klik tapi belum memiliki nilai.
Pada pertemuan berikutnya, kita akan membuatnya memiliki nilai/value agar bisa digunakan untuk operasi matematika



Challenge!

1. Cari image button start yang lain di internet dan gunakan pada game mu!
2. Ganti style font pada teks result dan question (font size, color)
3. Tambahkan properti font-family pada teks.

Contoh :

```
this.resultText = this.add.text(this.gameHalfWidth,  
    200, '0', {fontSize : '32px', fill: '#000',  
    fontFamily: 'Arial, sans-serif'})
```

Kamu bisa melihat pilihan font family disini: <tiny.cc/fontFamily>



Hari/Tanggal : _____

1. Sebutkan perintah apa saja yang dijalankan saat button Start di klik !

.....
.....
.....

2. Apa arti kode ini?

```
this.add.image(this.gameHalfWidth, this.gameHalfHeight  
+ 181, 'start-btn').setInteractive()
```

.....
.....
.....

3. Bagaimana strategi menambahkan image button number?

.....
.....
.....

4. Apa fungsi variabel heightDifference dan widthDifference?

.....
.....

5. Method apa yang digunakan untuk menambahkan button number?

.....
.....
.....



Timedoctor
Coding Academy

3

MATH FIGHTER: ADD NUMBER AND GENERATE QUESTION



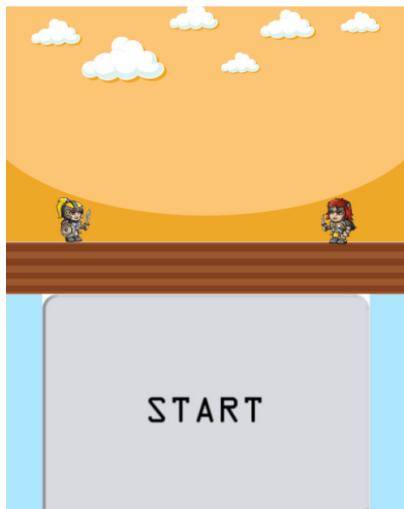
Apa Yang Akan Kita Pelajari?

1. Menambahkan Value Numbers
2. Menambahkan Value Question dan Operator



Math Fighter

Kali ini kita akan melanjutkan game Math Flighter dengan memberi nilai pada Buttons Numbers dan membuat Question !

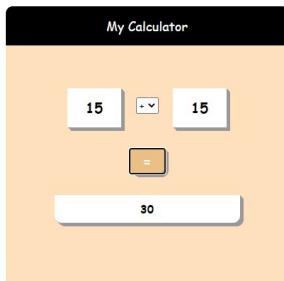


Button Numbers memiliki properti masing-masing. Sehingga ada 12 properti button yang sudah kita buat pada pertemuan sebelumnya. Setiap button juga harus bisa diklik agar dapat digunakan.

Masih ingat apa method yang ditambahkan untuk membuat gambar bisa diklik?

Selain itu, kita juga akan menambahkan **pertanyaan matematika menggunakan konsep yang sama dengan project Kalkulator!**

Ingin dengan project Kalkulator yang pernah kamu buat di Intermediate 1?





Hal pertama yang akan dilakukan adalah memberi nilai pada setiap button number.



Add Numbers Value

Untuk memberi nilai pada numbers, kita akan menggunakan konsep array pada method baru bernama **addNumbers()**.



Apa itu array?

Array disini akan digunakan untuk **menyimpan nilai numbers yang di klik** dan **mengambilnya menggunakan method getData()**

Pertama Inisialisasi properti ini pada method init() dulu ya!



Codes

```
this.numberArray = []
this.number = 0
```



1. Memberi Value Numbers

Buat kode ini pada method **addNumber()**



Codes

```
addNumber(pointer, object, event) {
  let value = object.getData('value')
}
```



Add Numbers Value

Variabel value digunakan untuk menyimpan data dengan key value, menggunakan method getData() untuk mengambil nilainya

Sekarang kita akan menggunakan konsep Array.
Ini adalah kode untuk **menghapus** dan **menambahkan** item pada Array.



Warm Up!

Sebelumnya kita sudah pernah mempelajarinya. Jika dbandingkan dengan Scratch :

Scratch

Javascript

add [thing] to [numbersArray]

this.numberArray.push()

delete [1] of [numbersArray]

this.numberArray.pop()

Kondisi yang akan ditambahkan dibagi menjadi 2 kondisi utama yaitu :

- Kondisi jika value nya bernilai bukan angka (yaitu button DEL dan button OK)
- Kondisi jika value nya bernilai angka (yaitu angka 1 sampai 0)



Nah sekarang, kita akan membuat kondisi untuk button OK dan button DEL terlebih dahulu

Lanjutkan kode ini pada method addNumber()



Codes

```
addNumber(pointer, object, event) {  
    let value = object.getData('value')  
  
    //tambahkan dari sini  
    if (isNaN(value)) {  
        if(value == 'del') {  
            this.numberArray.pop() //hapus array item terakhir  
            if(this.numberArray.length < 1) {  
                this.numberArray[0] = 0 //nilai indeks 0 = 0  
            }  
        }  
        if(value == 'ok') {  
            this.checkAnswer() // panggil method checkAnswer  
            this.numberArray = [] //kosongkan array  
            this.numberArray[0] = 0 //nilai indeks 0 = 0  
        }  
    } else {  
    }  
}
```

Pada kode di atas kita memakai kondisi di dalam kondisi karena button yang memiliki value “bukan angka” ada 2 yaitu DELETE dan OK.

Jika kamu menemukan error pada kode checkAnswer(), abaikan saja ya karena kita belum membuat methodnya.



Add Numbers Value

Selanjutnya kita tambahkan kondisi ELSE untuk membuat kondisi kedua yaitu **jika valuenya bernilai angka**

Lanjutkan kode ini setelah kondisi ELSE



Codes

```
} else {  
    //kondisi untuk button 0  
    if (this.numberArray.length == 1 &&  
        this.numberArray[0] == 0){  
        this.numberArray[0] = value  
    } else {  
        //jika panjang array kurang dari 10  
        if (this.numberArray.length < 10){  
            //tambah item ke array  
            this.numberArray.push(value)  
        }  
    }  
}
```

Pada kode di atas kita membuat 2 kondisi agar tidak bisa memasukkan lebih dari 1 angka dengan angka pertamanya adalah 0. Misalnya : 012

Selanjutnya tambahkan kode ini diluar kondisi tadi



Codes

```
addNumber(pointer, object, event) {  
    //buat dibawah penutup else  
    this.number = parseInt(this.numberArray.join(''))  
}
```

Kode `join('')` di atas berfungsi untuk **menyusun angka-angka yang terdapat pada numberArray** dan mengkonversinya menjadi string menggunakan `parseInt`



2. Menampilkan pada Teks

Kita baru saja mengatur kondisi untuk value dari setiap numbers, tapi kita belum memunculkannya pada teks.

Jadi sekarang ayo ikuti langkah ini agar value muncul pada teks!

Pertama pastikan kamu sudah Inisialisasi properti `this.resultText` = `undefined` pada method `init()` terlebih dahulu!

Lalu tulis kode ini di dalam method `addNumbers`, setelah kode `parseInt` tadi



Codes

```
this.resultText.setText(this.number)
const textHalfWidth = this.resultText.width * 0.5
this.resultText.setX(this.gameHalfWidth -
    textHalfWidth)
event.stopPropagation()
```



3. Panggil Method `addNumber()`

Untuk memunculkan teks tersebut, harus ada kondisi kapan value akan dimunculkan pada teks yaitu **saat button numbers di klik**

Tambahkan kode ini pada method `gameStart()`

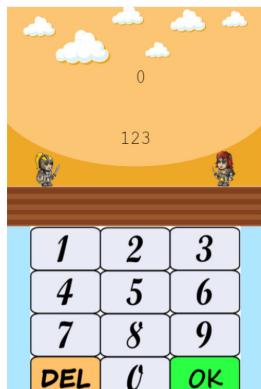


Codes

```
this.input.on('gameobjectdown', this.addNumber, this)
```

Pertemuan 3 – Add Number and Generate Question

Add Question



Sekarang coba jalankan game mu dan pastikan saat button numbers di klik sudah bisa muncul pada teks !



Add Question



1. Method getOperator()

Setelah berhasil membuat button Numbers dan memunculkannya pada teks, sekarang kita perlu membuat pertanyaannya.

Pertama kita perlu membuat method baru bernama **getOperator()** terlebih dahulu



Codes

```
getOperator ()  
{  
    const operator = ['+', '-', 'x', ':']  
    return operator[Phaser.Math.Between(0,  
        operator.length - 1)]  
}
```

Method **getOperator** digunakan untuk mengambil nilai array operator yang berisi item +, -, x, :



2. Method generateQuestion()

Selanjutnya kita membuat method untuk membuat pertanyaan matematika nya

Inisialisasi properti question dulu untuk menyimpan array question, seperti yang kita lakukan pada numberArray



Codes

```
this.question = []
```

Sama dengan project Kalkulator saat di Intermediate 1, kita akan membutuhkan variabel untuk menyimpan **angka pertama, angka kedua, dan operator**.

Buat method baru bernama generateQuestion()

Lalu tulis kode ini pada method generateQuestion()



Codes

```
generateQuestion() {  
    let numberA = Phaser.Math.Between(0, 50)  
    let numberB = Phaser.Math.Between(0, 50)  
    let operator = this.getOperator()  
  
    if (operator === '+') {  
        this.question[0] = `${numberA} + ${numberB}`  
        this.question[1] = numberA + numberB  
    }  
}
```

Add Question



Codes

```

if (operator === '-'){
    if (numberB > numberA) {
        this.question[0] = `${numberB} - ${numberA}`
        this.question[1] = numberB - numberA
    } else {
        this.question[0] = `${numberA} - ${numberB}`
        this.question[1] = numberA - numberB
    }
}

if (operator === 'x'){
    this.question[0] = `${numberA} x ${numberB}`
    this.question[1] = numberA * numberB
}

if (operator === ':'){
    do {
        numberA = Phaser.Math.Between(0, 50)
        numberB = Phaser.Math.Between(0, 50)
    }
    while (!Number.isInteger(numberA/numberB))

    this.question[0] = `${numberA} : ${numberB}`
    this.question[1] = numberA / numberB
}
}

```



Bisakah kamu melihat apa yang disimpan pada array question indeks ke-0 dan ke-1 ?



Contohnya:

```
this.question[0] = `${numberB} - ${numberA}`  
this.question[1] = numberB - numberA
```

Expression pertama artinya **question[0]** menyimpan pertanyaan,
Expression kedua artinya **question[1]** menyimpan hasilnya
Ini akan memudahkan saat nanti kita ingin melakukan check answer

Pada kode di tadi, kita juga menggunakan konsep perulangan **do while** pada kondisi pertanyaan pembagian



Apa itu do while?

Masih ingat apa perbedannya dengan while do?

```
do {  
    numberA = Phaser.Math.Between(0, 50)  
    numberB = Phaser.Math.Between(0, 50)  
}
```

Kode ini artinya game akan memberikan nilai pada numberA dan numberB antara 0 sampai 50 terlebih dahulu

```
while (!Number.isInteger(numberA/numberB))
```

Setelah itu, game akan mengecek jika numberA dibagi numberB bernilai integer,

```
this.question[0] = `${numberA} : ${numberB}`  
this.question[1] = numberA / numberB
```

Maka masukkan nilai numberA dibagi numberB pada array question.

Jika tidak, kembali ke perintah **do**



Pertemuan 3 – Add Number and Generate Question

Add Question

Selanjutnya, sama seperti resulText, kita harus menampilkan value array question pada teks

Pertama pastikan kamu sudah Inisialisasi properti `this.questionText` = `undefined` terlebih dahulu

Lalu tulis kode ini di dalam method generateQuestion, setelah penutup if



Codes

```
this.questionText.setText(this.question[0])
const textHalfWidth = this.questionText.width * 0.5
this.questionText.setX(this.gameHalfWidth -
    textHalfWidth)
```



3. Panggil Method generateQuestion()

Terakhir, kita perlu memanggil method generateQuestion()

Tulis kode ini di dalam method gameStart()



Codes

```
this.generateQuestion()
```

Selesai !

**Sekarang reload game mu dan pastikan question sudah muncul !
Pastikan juga semua button Numbers dan Delete dapat digunakan ya !**





Hari/Tanggal : _____

1. Apa nama array yang digunakan untuk menyimpan nilai numbers?

.....
.....
.....

2. Bagaimana kode untuk menambah dan menghapus item pada Array?

.....
.....
.....

3. Apa arti kode ini pada method addNumber?

```
if(value == 'del') {  
    this.numberArray.pop()
```

.....
.....
.....

4. Bisakah kita membuat angka 032 pada game ?

.....
.....
.....

5. Apa arti kode ini yang ditulis pada method gameStart()?
`this.input.on('gameobjectdown', this.addNumber, this)`

.....
.....
.....



Timedoctor
Coding Academy

4

MATH FIGHTER: CHECK ANSWER AND SPRITE HIT



Apa Yang Akan Kita Pelajari?

1. Menambahkan Player dan Enemy Attack
2. Menambahkan Sprite Overlaps and Hit



Math Fighter



Warm Up!

Apa saja yang sudah kamu buat di pertemuan sebelumnya?

Nah sedikit lagi game Math Fighter mu selesai lo!
Tapi sebelumnya, Slash belum bisa digunakan bukan?



Sekarang kita akan membuat slash dapat digunakan untuk menyerang,

Lalu kita buat game mu bisa mengecek jawaban agar Player bisa menyerang jika jawabannya benar dan Enemy bisa menyerang jika jawabannya salah

The screenshot shows a game interface. At the top, there is a subtraction equation: $13 - 8$. Below the equation, the result 0 is displayed. At the bottom, there is a numeric keypad with a grid of numbers from 1 to 9, a **DEL** button, and an **OK** button.

| | | |
|------------|---|-----------|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |
| DEL | 0 | OK |



Player and Enemy Attack

Agar Player maupun Enemy bisa menyerang dengan Slash, kita harus periksa dulu apakah jawabannya benar atau salah.

Untuk itu kita harus buat method baru yang bertugas memeriksa jawaban



1. Method checkAnswer()

Inisialisasi properti **correctAnswer** dulu pada method **init()**



Codes

```
this.correctAnswer = undefined
```

Selanjutnya buat method baru bernama **checkAnswer()**



Codes

```
checkAnswer ()  
{  
    if (this.number == this.question[1])  
    {  
        this.correctAnswer = true  
    } else {  
        this.correctAnswer = false  
    }  
}
```

Kode di atas memeriksa jika jawabannya benar, maka **correctAnswer** bernilai True



Pertemuan 4 - Check Answer and Sprite Hit Player and Enemy Attack

Selanjutnya kita akan membuat kondisi pada update, untuk menentukan perintah apa yang diberikan pada Player dan Enemy



2. Attack Animation

Inisialisasi properti ini pada method init()



Codes

```
this.playerAttack = false  
this.enemyAttack = false
```

Selanjutnya buat kode ini pada method update(time)



Codes

```
update(time)  
{  
    if(this.correctAnswer == true && !this.playerAttack)  
    {  
        this.player.anims.play('player-attack', true)  
        this.time.delayedCall(500, () => {  
            this.createSlash(this.player.x + 60,  
                            this.player.y, 0, 600)  
        })  
        this.playerAttack = true  
    }  
}
```



Codes

```
if(this.correctAnswer == undefined)
{
    this.player.anims.play('player-standby', true)
    this.enemy.anims.play('enemy-standby', true)
}

if(this.correctAnswer == false && !this.enemyAttack)
{
    this.enemy.anims.play('enemy-attack', true)
    this.time.delayedCall(500, () => {
        this.createSlash(this.enemy.x - 60,
                        this.enemy.y, 2, -600)
    })
    this.enemyAttack = true
}
}
```

Kode di atas terdiri dari 3 kondisi

Kondisi 1 -> jika jawaban benar maka Player akan beranimasi attack dan menyerang ke arah kanan layout (enemy)

Kondisi 2 -> jika belum menjawab, maka Player dan Enemy beranimasi standby

Kondisi 3 -> jika jawaban salah, maka Enemy akan beranimasi attack dan menyerang ke arah kiri layout (player)

Selanjutnya kita akan membuat method createSlash() yang sudah dipanggil di atas



Pertemuan 4 - Check Answer and Sprite Hit Player and Enemy Attack

Buat method `createSlash()` berisi parameter seperti kode ini



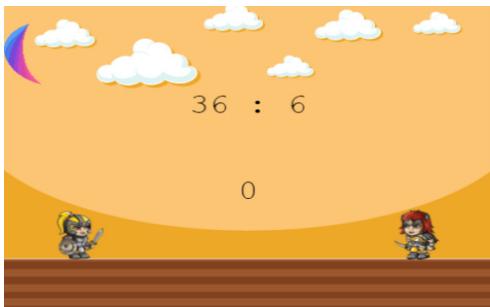
Codes

```
createSlash(x, y, frame, velocity, flip = false)
{
    this.slash.setPosition(x, y)
        .setActive(true)
        .setVisible(true)
        . setFrame(frame)
        .setVelocityX(velocity)
        .setFlipX(flip)
}
```

Sekarang coba tes pada game mu apakah slash sudah muncul setelah menjawab pertanyaan dengan benar maupun salah!



Apakah Slash menjadi seperti ini?



Itu terjadi karena kita belum membuat Slash Overlaps dengan Player ataupun Enemy

Nah sekarang kita akan membuat agar Player ataupun Enemy bisa overlaps dengan Slash lalu Slash destroy!



Sprite Overlaps and Hit

Untuk membuat Overlaps Slash dan Player maupun Slash dan Enemy, kita akan memerlukan method baru yang dipanggil setelah terjadi Overlaps, yaitu method `spriteHit()`.

Method `spriteHit()` menampung perintah untuk **animasi Player Hit dan Enemy Hit, serta memanggil method generateQuestion()**.

Buat method `spriteHit()` dan tulis kode ini



Codes

```
spriteHit(slash, sprite)
{
    slash.x = 0
    slash.y = 0
    slash.setActive(false)
    slash.setVisible(false)

    if(sprite.texture.key == 'player') {
        sprite.anims.play('player-hit', true)
    } else {
        sprite.anims.play('enemy-hit', true)
    }

    this.time.delayedCall(500, () => {
        this.playerAttack = false
        this.enemyAttack = false
        this.correctAnswer = undefined
        this.generateQuestion()
    })
}
```



Pertemuan 4 - Check Answer and Sprite Hit Player and Enemy Attack

Selanjutnya adalah menambahkan kode Overlaps



Do it Your Self !

Coba kamu buat sendiri kode overlaps ya!

Buat 2 kode overlaps :

1. Overlaps Player dan Slash lalu panggil method spriteHit()
2. Overlaps Enemy dan Slash lalu panggil method spriteHit()

Coba jalankan dan lihat hasilnya ya!

- ✓ Apakah animasi Player Attack dan Enemy Attack sudah terlihat saat akan menyerang?
- ✓ Apakah Slash hilang setelah menyentuh Player atau Enemy?
- ✓ Apakah animasi Player Hit dan Enemy Hit sudah terlihat setelah overlaps dengan Slash ?
- ✓ Apakah pertanyaan selalu muncul lagi setelah selesai menjawab?



Hari/Tanggal: _____

1. Method apa yang digunakan untuk memeriksa jawaban yang benar?

.....
.....
.....

2. Apa saja perintah yang dijalankan jika jawaban benar?

.....
.....

3. Pada method spriteHit() terdapat kode berikut ini. Coba kamu isi titik-titik yang kosong!

```
this.(.....).delayedCall(500, () => {
    this.playerAttack = false
    this.(.....) = false
    .....).correctAnswer = undefined
    this.(.....)()
```

}

4. Bagaimana kode untuk overlaps Player dan Slash?

.....
.....

5. Bagaimana kode untuk overlaps Enemy dan Slash ?

.....
.....
.....



Timedoctor
Coding Academy

5

MATH FIGHTER: SCORE AND TIMER



Apa Yang Akan Kita Pelajari?

1. Menambahkan Score
2. Menambahkan Timer
3. Menambahkan Game Over Scene



Math Fighter



Bagaimana perkembangan game mu?
Nah hari ini kita akan menambahkan challenge agar game mu makin seru !

**Seperti yang telah disebutkan di awal, Math Fighter akan memiliki Timer.
Jika waktu habis, maka scene akan berpindah ke scene Game Over dan permainan selesai**



Bisakah kamu menyebutkan contoh game yang menggunakan Timer?



Selain Timer, kita juga akan menambahkan Score yang bisa bertambah dan berkurang tergantung jawaban Player.

Yuk kita buat game Math Fighter memiliki Score dan Timer!



Add Score

Kita juga akan menambahkan Score pada game ini. Score akan bertambah jika jawaban benar dan berkurang jika jawaban salah.

Sebelum itu, ayo kita buat Score label untuk menampilkan teks Score dulu !



Masih ingat cara membuat Score Label?
Caranya sama loh dengan Score label saat di game Corona Buster!



Pada folder src, buat folder ui dan tambahkan file baru bernama ScoreLabel.js

Codes

```
import Phaser from 'phaser'

const formatScore = (gameScore) => `Score: ${gameScore}`

export default class ScoreLabel extends Phaser.GameObjects.Text
{
    constructor(scene, x, y, skor, style)
    {
        super(scene, x, y, formatScore(skor), style)
        this.score = skor
    }

    setScore(skor)
    {
        this.score = skor
        this.setText(formatScore(this.score))
    }

    getScore()
    {
        return this.score
    }

    add(points)
    {
        this.setScore(this.score + points)
    }
}
```



Sekarang kembali ke MathFighterScene dan lakukan langkah berikut ini ya !

Buat method baru bernama `createScoreLabel()`



Codes

```
createScoreLabel(x, y, score)
{
    const style = { fontSize: '24px', fill: '#000',
        fontStyle: 'bold' }
    const label = new ScoreLabel(this, x, y, score,
        style).setDepth(1)

    this.add.existing(label)
    return label
}
```

Selanjutnya kita perlu import kelas ScoreLabel dan membuat nilai score bisa bertambah atau berkurang



Do it Your Self !

Coba lakukan sendiri dengan ikuti petunjuk ini ya!

- Import kelas ScoreLabel pada MathFighterScene.js
- Inisialisasi `this.scoreLabel = 0`
- Panggil method `createScoreLabel()` di method `create()`
`this.scoreLabel = this.createScoreLabel(26, 16, 0)`



Untuk membuat score bertambah atau berkurang, lihat pada method spriteHit() tepatnya pada kondisi if else ini.

Masukkan kode yang bertanda komentar

Codes

```
if(sprite.texture.key == 'player') {  
    sprite.anims.play('player-hit', true)  
    //masukkan 2 baris kode di bawah ini  
    if(this.scoreLabel.getScore() > 0){  
        this.scoreLabel.add(-50)  
    }  
} else {  
    sprite.anims.play('enemy-hit', true)  
    //masukkan 1 baris kode di bawah ini  
    this.scoreLabel.add(50)  
}
```

Nah sekarang Score sudah terlihat dan pastikan nilainya sudah bisa berubah-ubah sesuai kondisi jawaban salah atau benar ya!





Add Timer

Game mu sudah hampir selesai ! Agar lebih seru ayo kita tambahkan kondisi Game Over !
Jika waktu sudah habis maka akan muncul layout Game Over.



1. Menambahkan Timer

Pertama kita tambahkan timer terlebih dahulu. Pada game ini Timer akan mulai menghitung mundur (countdown) saat button Start ditekan.

Inisialisasi properti ini pada method init()



Codes

```
this.timerLabel = undefined  
this.countdownTimer = 60
```

Lalu tampilkan teks score di method create()



Codes

```
this.timerLabel = this.add.text(this.gameHalfWidth, 16,  
null).setDepth(5)
```

Selanjutnya kita harus menambahkan kondisi pada method update() "Jika start game dipanggil maka munculkan Timer"

Tambahkan kode berikut ini di dalam method update()



Codes

```
if(this.startGame = true){
    this.timerLabel.setStyle({
        fontSize: '24px',
        fill : '#000',
        fontStyle: 'bold',
        align : 'center'
    }).setText(this.countdownTimer)
}
```



Apakah timer sudah muncul?

Tapi timer belum menghitung mundur bukan?

Untuk itu kita perlu menambahkan **expression aritmatika untuk mengurangi nilai countdownTimer setiap detik**

Pertama, buat dulu method baru bernama gameOver()



Codes

```
gameOver()
{
    this.countdownTimer -= 1
    if (this.countdownTimer < 0) {
        this.scene.start('game-over-scene', {score:
            this.scoreLabel.getScore() })
    }
}
```

Selanjutnya, Inisialisasi properti ini pada method init()



Codes

```
this.timedEvent = undefined
```



Pertemuan 5 – Score and Timer

Add Timer

Tambahkan kode ini di dalam method gameStart()



Codes

```
this.timedEvent = this.time.addEvent({  
    delay: 1000,  
    callback: this.gameOver,  
    callbackScope: this,  
    loop: true  
})
```

properti **timedEvent** menggunakan Timer Event yang berisi kode **delay (wait) selama 1 detik sebelum memanggil method gameOver() yang dijalankan berulang-ulang.**

Lalu pada method **gameOver()** terdapat kode :

`this.countdownTimer -= 1` (kurangi 1) sampai nilainya kurang dari 0, maka pindah ke scene Game Over.

Sekarang jalankan game mu dan cek apakah timer sudah bisa menghitung mundur?

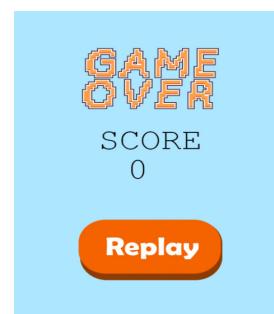
Kita juga akan membuat layout berpindah ke layout Game Over



Do it Your Self !

Buat layout game over mu sendiri ya! Kamu bisa menggunakan gambar pada game sebelumnya atau mencari gambar mu sendiri di Google!

Tampilkan juga score pada layout Game Over ya!





Hari/Tanggal : _____

1. Pada method `createScoreLabel()` bagaimana kode untuk menggunakan kelas `ScoreLabel` ?

.....
.....
.....

2. Apa arti kode berikut ini?

```
if(this.scoreLabel.getScore() > 0){  
    this.scoreLabel.add(-50)
```

.....
.....
.....

3. Properti apa yang digunakan untuk menyimpan waktu selama 60 detik?

.....
.....
.....

4. Apa yang akan terjadi dengan menambahkan kode ini `setText(this.countdownTimer)` ?

.....
.....
.....

5. Method apa yang dipanggil setiap delay selama 1 detik ?

.....
.....
.....



Timedoctor
Coding Academy

6

MEMORY GAME: ADDING BOXES



Apa Yang Akan Kita Pelajari?

1. Menambahkan Box
2. Menambahkan Player
3. Membuat Animasi Player



Memory Game

Kali ini kita akan mulai membuat **Memory Game**.

Memory Game memiliki konsep coding yang mirip dengan Match Game pada Intermediate 1.

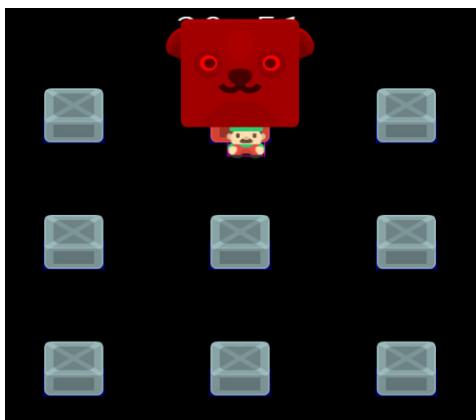


Masih ingat dengan Match Game?

Pada Match Game, cara bermainnya adalah dengan cara klik pada Card untuk menemukan kartu yang sama



Tetapi pada Memory Game kita akan **menggunakan Player untuk membuka Box dan mengecek Karakter Binatang yang sama**.



Salah satu kotak berisi **karakter Beruang** yang **tidak memiliki pasangan** dan merupakan **Box jebakan**.

Selain itu, Memory Game juga akan memiliki **Timer**, game akan jadi lebih menantang!



Nah seperti biasa, hal pertama yang akan kita lakukan adalah melakukan persiapan template



Preparation

1. Prepare Template

Lakukan Persiapanmu sendiri ya!



Do it Your Self !

Kamu bisa menandai bagian yang sudah dikerjakan agar tidak bingung !

- Copy paste** Template Phaser dan **Rename** menjadi *Memory Game*
- Download assets di : tiny.cc/memorygame_assets
- Buat Folder **Public > Images > Masukkan Semua Assets**
- Pada **Index.html**, Ubah Judul Tab menjadi *Memory Game*
- Pada **Folder scenes**, Buat file *MemoryGameScene.js*
- Pada **main.js, Konfigurasi** *MemoryGameScene.js*
- Import MemoryGameScene** dan tambahkan pada **scene**
- Atur ukuran layout game, **width : 720** dan **height : 680**, serta **gravity : 0**
- Tambahkan **properti Scale** pada main.js

```
scale: {  
    mode: Phaser.Scale.FIT,  
    autoCenter: Phaser.Scale.CENTER_BOTH  
},
```



2. Struktur Kelas MemoryGameScene

Selanjutnya buatlah struktur kelas MemoryGameScene.js



Codes

```
import Phaser from 'phaser'  
export default class MemoryGameScene extends Phaser.  
Scene  
{  
    constructor()  
    {  
        super('memory-game-scene')  
    }  
    init()  
    {  
    }  
    preload()  
    {  
    }  
    create()  
    {  
    }  
    update()  
    {  
    }  
}
```

Selanjutnya kita perlu memuat assets yang kita punya pada method preload().



3. Load Assets

Ikuti petunjuk di bawah ini agar key yang digunakan sama.

| IMAGE/ SPRITE SHEET | KEY | FRAME WIDTH |
|------------------------|-------------|----------------|
| bg.jpg | 'bg' | - |
| sokoban_tilesheet.png | 'tilesheet' | 64 |
| chicken.png | 'chicken' | - |
| duck.png | 'duck' | - |
| bear.png | 'bear' | - |
| parrot.png | 'parrot' | - |
| penguin.png | 'penguin' | - |
| play.png | 'play' | - |



4. Create Background

Selanjutnya adalah create background. Seperti biasa agar lebih mudah menambahkan gambar, ayo kita buat properti untuk menyimpan posisi tengah game.

Buat kode ini pada method init()



Codes

```
this.halfWidth = this.scale.width / 2
this.halfHeight = this.scale.height / 2
```

Pertemuan 6 – Adding Boxes Preparation



Selanjutnya tambahkan background dengan kode ini pada method create()

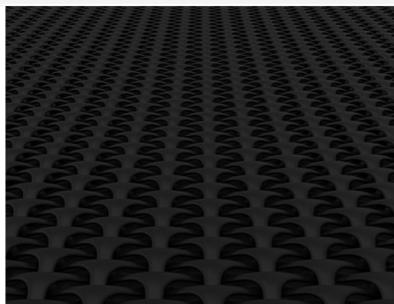


Codes

```
this.add.image(this.halfWidth, this.halfHeight, 'bg')
.setScale(3)
```

Lihat hasilnya dan pastikan terlihat seperti ini ya !

localhost:8000



Kamu bebas menggunakan background yang kamu suka ya !

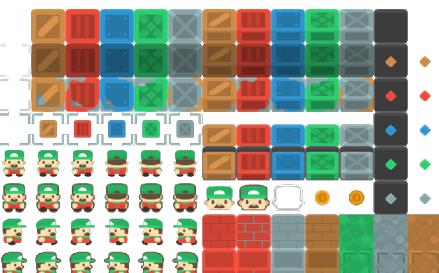


Adding Boxes

Kita akan menggunakan box yang terdapat pada tilesheet ini.

Box yang akan kita buat adalah sebanyak 6 box.

Jadi kita akan mengelompokkannya menjadi grup dan menampilkan dengan konsep loop.





Yuk ikuti langkah berikut ini!

Pertama, buat properti boxGroup dulu pada method init()



Codes

```
this.boxGroup = undefined
```

Selanjutnya kita akan menginisialisasinya kembali pada method create sebagai static objek (tidak bergerak)



Codes

```
this.boxGroup = this.physics.add.staticGroup()
```

Nah, untuk menampilkan box kita akan menggunakan konsep **for loop**.



Kenapa menggunakan for loop bukannya iterasi seperti game sebelumnya?

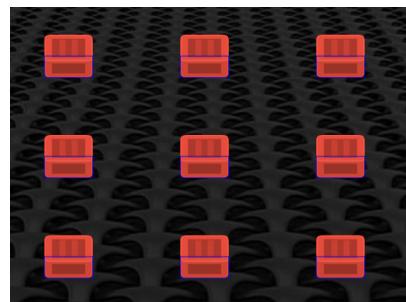
Itu karena kita **sudah tahu berapa kali akan melakukan perulangan**.

Dengan kata lain, **jumlah box yang dimunculkan sudah ditentukan yaitu 9 box**.



Bagaimana caranya?

Yuk lihat kode berikut ini!



Pertemuan 6 – Adding Boxes Preparation



Buatlah method baru bernama `createBoxes()` berisi kode ini



Codes

```
createBoxes()
{
    const width = this.scale.width
    let xPer = 0.25
    let y = 150
    for (let row = 0; row < 3; row++)
    {
        for (let col = 0; col < 3; col++)
        {
            this.boxGroup.get(width * xPer, y, 'tilesheet', 7)

            xPer += 0.25
        }

        xPer = 0.25
        y += 150
    }
}
```

Perhatikan kode diatas, ada 2 perulangan for loop bukan?
Pertama, for loop digunakan untuk **mengatur barisan box secara vertikal (row)**

```
for (let row = 0; row < 3; row++)
{
    //ada for loop yg kedua
    xPer = 0.25
    y += 150
}
```

- **startValue** = 0
- **stopValue** = 3
- **increment** = 1

Artinya perulangan akan dilakukan 3 kali



Lalu kode for loop di dalamnya digunakan untuk **mengatur box secara horizontal (column) dan menampilkan box menggunakan method get()**

Perhatikan kode ini!

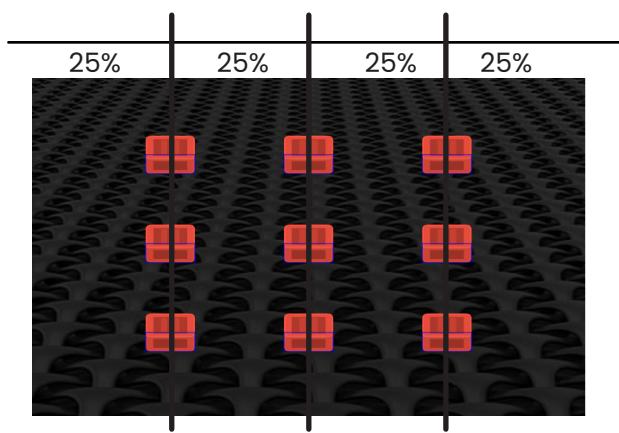
```
for (let col = 0; col < 3; col++)  
{  
    this.boxGroup.get(width * xPer, y, 'tilesheet', 7)  
    xPer += 0.25  
}
```



Kenapa pada method get() kita menentukan posisi x box = width * xPer ??

Sebelumnya, **width** didefiniskan dengan nilai **this.scale.width** **yaitu bernilai 720**.

Lalu **xPer bernilai 0.25**, atau yang berasal dari **25%**. Sehingga bisa digambarkan menjadi seperti ini



Lalu pada kode **xPer += 0.25**, setiap perulangan nilai xPer ditambah dengan 0.25 sehingga memberikan jarak yang seimbang sampai perulangan terakhir.



Begini juga dengan kode `y += 150` artinya setiap pengulangan di baris berikutnya, nilai y akan ditambah 150

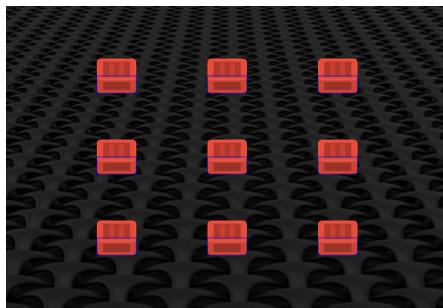
Selanjutnya panggil method tadi pada create()



Codes

```
this.createBoxes()
```

Lihat hasilnya dan pastikan sudah seperti ini ya!



Challenge!

Untuk menguji apa yang dihasilkan oleh kode for loop (column), Coba kamu jadikan komentar pada bagian for loop (row) dan penutupnya seperti ini

```
//for (let row = 0; row < 3; row++)
//{
  for (let col = 0; col < 3; col++)
  {
    this.boxGroup.get(width * xPer, y, 'tilesheet', 7)

    xPer += 0.25
  }
  xPer = 0.25
  y += 150
}
//}
```



Adding Player

Sekarang ayo kita tambahkan Player dan buat Player agar bisa digerakkan dengan key arrow dan beranimasi.

Pertama, buat properti player dan cursor pada method init()



Codes

```
this.player = undefined  
this.cursors = this.input.keyboard.createCursorKeys()
```

Selanjutnya, buat method baru bernama createPlayer() dan tulis kode ini di dalamnya



Codes

```
const player = this.physics.add.sprite(this.halfWidth,  
    this.halfHeight + 30, 'tilesheet')  
    .setSize(40, 16).setOffset(12, 38)  
player.setCollideWorldBounds(true)  
  
this.anims.create({  
    key: 'standby',  
    frames: [{ key: 'tilesheet', frame: 52 }]  
})  
  
this.anims.create({  
    key: 'down',  
    frames: this.anims.generateFrameNumbers('tilesheet',  
        { start: 52, end: 54 }),  
    frameRate: 10,  
    repeat: -1
```

Pertemuan 6 – Adding Boxes Adding Player



Nah kode di atas hanya menambahkan 2 animasi yaitu standby dan down. Sekarang coba kamu tambahkan sendiri animasi lainnya dengan petunjuk ini ya!



Do it Your Self !

1. Animasi up

key : 'up'
frame start : 55, end : 57

2. Animasi left

key : 'left'
frame start : 81, end : 83

3. Animasi right

key : 'right'
frame start : 78, end : 80

Terakhir **tambahkan kode return player** paling bawah



1. Move Player

Nah selanjutnya ayo kita buat Player bisa digerakkan dengan **key arrow**. Caranya sama dengan game-game sebelumnya ya!

Buat method movePlayer() dengan parameter player



Codes

```
movePlayer(player) {  
    const speed = 200  
}
```



Nah, kode untuk menggerakkan Player kamu buat sendiri sesuai contoh dan petunjuk ini ya!



Do it Your Self !

Di bawah kode `const speed = 200` buat kode seperti ini:

```
if (this.cursors.left.isDown)
{
    this.player.setVelocity(-speed, 0)
    this.player.anims.play('left', true)
}
```

Buat juga kondisi else if untuk right, up, down, dan standby

1. **right** -> `setVelocity(speed, 0)`
2. **up** -> `setVelocity(0, -speed)`
3. **down** -> `setVelocity(0, speed)`
4. **standby** -> `setVelocity(0, 0)`

Selanjutnya panggil method `movePlayer()` pada `update()`



Codes

```
this.movePlayer(this.player)
```

Terakhir panggil method `createPlayer()` pada `create()`



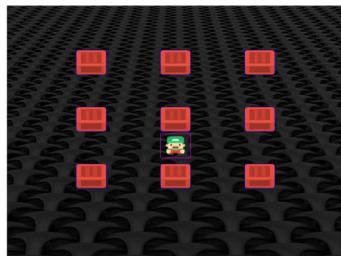
Codes

```
this.player = this.createPlayer()
```

Pertemuan 6 – Adding Boxes Adding Player

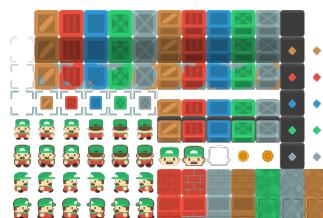


Sekarang lihat hasilnya yaa !
Apa Playernya sudah muncul dan bisa
digerakkan?
Apa animasinya sudah berjalan
dengan baik?



Challenge!

Lihat pada gambar
sokoban_tilesheet mu!



Coba kamu pilih frame box yang lain dan ubah nomor
frame (angka 7) pada kode `this.boxGroup.get(width *
xPer, y, 'tilesheet', 7)` di method `createBox()`



Hari/Tanggal : _____

1. Bagaimana kode untuk menambahkan boxGroup sebagai static object di method create() ?

.....
.....
.....

2. Mengapa menggunakan perulangan for loop untuk menampilkan box ?

.....
.....
.....

3. Jika kode loopnya seperti ini berapa box yang akan muncul?

```
for (let col = 1; col < 4; col++)
```

.....
.....

4. Apa fungsi variabel xPer ?

.....
.....
.....

5. Apa perbedaan kode setVelocity() pada kode menggerakkan Player ke atas, bawah, kanan, dan kiri?

.....
.....
.....



Timedoctor
Coding Academy

7

MEMORY GAME: OPEN THE BOX



Apa Yang Akan Kita Pelajari?

1. Menambahkan Collider Player dan Box
2. Membuat Efek Box Disentuh Player
3. Membuat Box Terbuka



Memory Game



Warm Up!

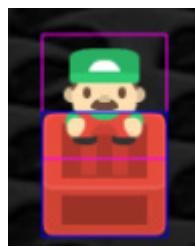


Masih ingat bagaimana cara menambahkan box?
Konsep loop apa yang digunakan?

Nah setelah berhasil menambahkan Box, coba kamu gerakkan Player ke arah box !



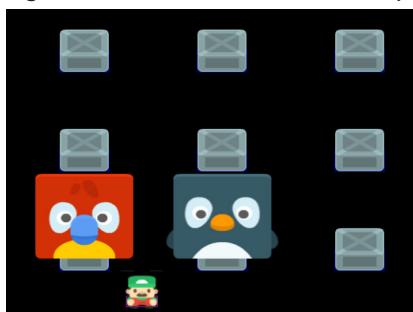
Apakah Player menembus Box?
Menurutmu apakah sebaiknya Player bisa menembus Box atau tidak?



Kali ini kita akan membuat **Player tidak bisa menembus Box (collides)**, karena Box di dunia nyata itu adalah benda padat dan tidak dapat ditembus.

Jadi ayo kita buat seperti nyata ya!

Selain itu, kita akan membuat Box memiliki efek sentuhan, serta **dapat dibuka dengan Keyboard Spasi** dan akan memunculkan **Item Binatang** yang sudah kita load sebelumnya.





Player and Box Collides



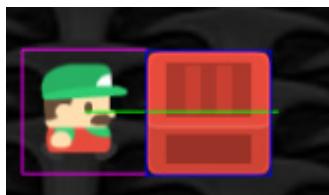
Kita sudah sering menggunakan kode Collider bukan?
Bagaimana kode untuk menambahkan Collider?

Sekarang coba kamu tambahkan collider sendiri ya !



Do it Your Self !

Tambahkan Collider untuk `this.player` dan `this.boxGroup` pada method `create()` !



Jika sudah, pastikan kode mu berhasil ya !

Coba gerakkan lagi Player menuju Box dan pastikan Player tidak bisa menembusnya !



Tapi coba gerakkan Player menuju Box. Player bisa menembus box dari depan bukan?



Seharusnya juga Player bisa berada di belakang Box bukan?





Nah untuk mengatasi itu kita perlu menambahkan kode untuk mengatur **depth(layering)** dari Box agar Player tidak menembus Box dan mengatur size dan offset agar Player seolah-olah berada di belakang Box.

Tambahkan kode ini di dalam method update()



Codes

```
this.children.each(c => {
  /** @type {Phaser.Physics.Arcade.Sprite} */
  // @ts-ignore
  const child = c

  child.setDepth(child.y)
})
```

Simpan projectmu dan coba gerakkan Player ke bawah Box !

Patikan Player tidak menembus Box lagi ya !



Nah selanjutnya adalah membuat Player bisa berada di belakang Box.

Pergi ke method createBoxes() dan tambahkan kode bertanda komentar



Codes

```
this.boxGroup.get(width * xPer, y, 'tilesheet', 7)
//tambahkan kode di bawah ini
.setsize(64, 32)
.setOffset(0, 32)
```

Coba gerakkan Player ke belakang Box dan pastikan hasilnya seperti ini !





Apakah 6 box mu menghilang?

itu disebabkan oleh kode :

```
child.setDepth(child.y)
```



Kode itu berfungsi agar **urutan layering ditentukan oleh di titik y berapa objek di create**. Semakin kecil nilai y maka objek berada semakin belakang.

Untuk itu ayo kita ubah kode menambahkan **background** di method **create()** menjadi seperti ini



Codes

```
this.add.image(this.halfWidth, 150, 'bg')  
.setScale(3)
```



Handle Active Box

Active Box adalah properti yang berperan penting untuk menyimpan Box yang disentuh atau dibuka oleh Player.

Nah pertama kita akan membuat **Box berubah warna saat di sentuh oleh Player (efek sentuhan)**.

Namun selain itu juga kita perlu **mengembalikan Box ke warna semula saat Player sudah tidak menyentuhnya**



Bagaimana caranya mengubah warna?

Kita akan memanfaatkan frame yang lain menggunakan kode **setFrame()** !

Pertama, inisialisasi dulu properti activeBox pada method init()



Codes

```
this.activeBox = undefined
```



Selanjutnya ayo kita buat method baru bernama **handlePlayerBoxCollide()**



Codes

```
handlePlayerBoxCollide(player, box) {
    if (this.activeBox) {
        return
    }
    this.activeBox = box
    this.activeBox.setFrame(9)
}
```

this.activeBox digunakan untuk **menyimpan nilai box yang disentuh oleh Player**.

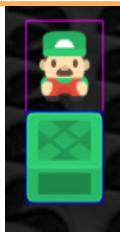
Kemudian frame akan diganti menjadi warna hijau (frame 9), Kamu juga bisa mengganti dengan frame lain yang kamu suka!

Lalu panggil method tadi pada kode collider di method **create()** menjadi seperti ini



Codes

```
this.physics.add.collider(this.player, this.boxGroup,
    this.handlePlayerBoxCollide, undefined, this)
```



Sekarang coba gerakkan lagi Player mu ke salah satu box, apakah box itu sudah berubah warna ?



Kode itu hanya memberi efek sentuhan pada box pertama saja bukan?

Nah sekarang ayo kita buat agar semua box bisa menunjukkan efek sentuhan dan membuat efek itu hilang setelah Player bergerak menjauh !



1. Update Active Box

Kita perlu membuat method baru bernama updateActiveBox()

Tulislah kode ini pada method updateActiveBox()



Codes

```
updateActiveBox()
{
    if (!this.activeBox)
    {
        return
    }
    const distance = Phaser.Math.Distance.Between(
        this.player.x, this.player.y,
        this.activeBox.x, this.activeBox.y
    )

    if (distance < 64)
    {
        return
    }
    this.activeBox.setFrame(7)//mengembalikan frame merah
    this.activeBox = undefined
}
```



Selanjutnya panggil method tersebut pada method update()



Codes

```
this.updateActiveBox()
```

Coba lihat hasilnya dan pastikan
Box hanya berubah warna jika
Player menyentuhnya ya!



 Bagaimana logikanya ?
Pada kode itu kita mengukur terlebih dahulu jarak antara Player dengan Box. Jika nilainya kurang dari 64, maka box itu akan berubah ke frame 9 (berwarna hijau).

Bisakah kamu melihat lebih detail kode yang mana yang memiliki fungsi seperti itu?

Coba perhatikan kode ini dan isi titik-titik dengan deskripsi kodenya ya!

```
const distance = Phaser.Math.Distance.Between(
    this.player.x, this.player.y,
    this.activeBox.x, this.activeBox.y
)
```

Apa fungsi kode di atas?



```
if (distance < 64)
{
    return
}
```

Apa fungsi kode di atas?

Nah sedangkan untuk kode di baris terakhir yaitu

```
this.activeBox = undefined
```

Artinya adalah mengosongkan kembali nilai activeBox (kembali ke warna merah) karena jarak Player dan Box lebih dari 64.

Kamu sudah berhasil membuat efek sentuhan pada Box ! Selanjutnya kita akan membuat Box dapat terbuka saat menekan keyboard Spasi.



2. Open Box

Sebelum membuat Box bisa dibuka, ayo kita atur dulu posisi Binatang pada setiap Box.

Buatlah array baru di bawah kode import Phaser seperti ini.



Codes

```
const level = [
    [1, 0, 3],
    [2, 4, 1],
    [3, 4, 2]
]
```

Open Box



Array level digunakan untuk menyimpan data Binatang yang ada di dalam Box.



Apa maksud angka-angka di dalamnya?

Angka atau Indeks adalah tanda untuk setiap Item Binatang

| | |
|-------------|--|
| Indeks ke 0 | |
| Indeks ke 1 | |
| Indeks ke 2 | |
| Indeks ke 3 | |
| Indeks ke 4 | |

Jika digambarkan maka array level berisi nilai seperti urutan gambar di bawah ini



Nah kita akan membuat kode dari ilustrasi ini pada method `openBox()` menggunakan konsep **Switch Case**.



Vocabulary

Switch Case adalah statement yang digunakan untuk menampilkan aksi yang berbeda pada beberapa kondisi berbeda.



1. Add New Key and Property

Key itemType adalah nama yang diberikan pada kelompok data di dalam array level. Key itemType akan digunakan (get) pada **variabel itemType** yang menyimpan nilai Binatang yang dipilih.

Pertama, tambahkan dulu kode `setData()` pada method `createBox()` seperti pada tanda comment.



Codes

```
this.boxGroup.get(width * xPer, y, 'tilesheet', 7)
//tambahkan kode di bawah ini
.setData('itemType', level[row][col])
```

Selanjutnya kita akan memerlukan **properti itemsGroup** untuk menyimpan group Binatang.

Inisialisasi `itemsGroup` pada method `init()`



Codes

```
this.itemsGroup = undefined
```

Lalu tambahkan `itemsGroup` pada method `create` sebagai **object group**.



Codes

```
this.itemsGroup = this.add.group()
```

Selanjutnya kita akan membuat **method openBox** sebagai **method yang mengatur seluruh kode yang diperlukan untuk membuka Box** atau sama dengan **menampilkan item Binatang di setiap Box**.



2. Method openBox

Buatlah method baru bernama **openBox()** dengan parameter **box**. Tulis kode ini di dalamnya.



Codes

```
openBox(box)
{
    if(!box) {
        return
    }
    const itemType = box.getData('itemType')
    let item

    switch (itemType)
    {
        case 0:
            item = this.itemsGroup.get(box.x, box.y)
            item.setTexture('bear')
            break

        case 1:
            item = this.itemsGroup.get(box.x, box.y)
            item.setTexture('chicken')
            break

        case 2:
            item = this.itemsGroup.get(box.x, box.y)
            item.setTexture('duck')
            break
    }
}
```



Codes

```

    case 3:
        item = this.itemsGroup.get(box.x, box.y)
        item.setTexture('parrot')
        break

    case 4:
        item = this.itemsGroup.get(box.x, box.y)
        item.setTexture('penguin')
        break
    }
    if (!item) {
        return
    }

    box.setData('opened', true)
}

```

Selanjutnya tambahkan kode berikut ini di dalam method `handlePlayerBoxCollide()` baris paling bawah



Codes

```

const opened = box.getData('opened')

if (opened)
{
    return
}

```

Kode yang baru ditambahkan berfungsi untuk mengambil key '`opened`'. Kemudian melakukan pengecekan, Jika bernilai true maka return (tetap terbuka/binatang muncul).



Nah terakhir ayo kita tambahkan kode agar saat keyboard Space di klik, maka gambar Binatang akan muncul

Tambahkan kode ini di dalam method movePlayer() di baris paling bawah

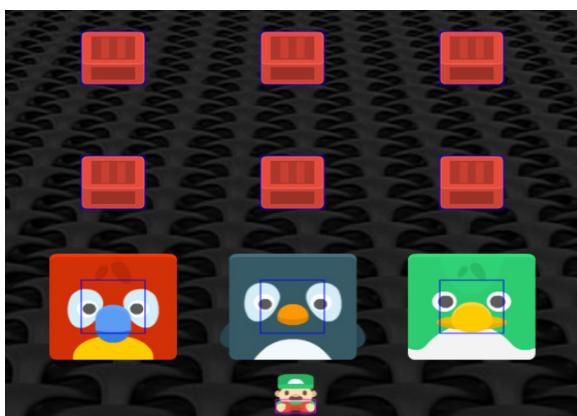


Codes

```
const spaceJustPressed = Phaser.Input.Keyboard.  
JustUp(this.cursors.space)

if (spaceJustPressed && this.activeBox)  
{  
    this.openBox(this.activeBox)  
  
    this.activeBox.setFrame(7)  
    this.activeBox = undefined  
}
```

Nah sekarang Box sudah dapat memunculkan Item Binatang ! Coba kamu lihat hasilnya dan pastikan semua Box sudah bisa dibuka ya !





Hari/Tanggal : _____

1. Bagaimana kode untuk menambahkan collider Box dan Player?

.....
.....
.....

2. Method apa saja yang baru ditambahkan?

.....
.....

3. Kelas Phaser apa yang digunakan untuk mengukur jarak Player dan Box?

.....
.....
.....

4. Apa itu Switch Case statement? Berapa Case yang dibuat pada method openBox?

.....
.....
.....

5. Bagaimana kode untuk mendeklarasikan variabel spaceJustPressed ?

.....
.....
.....



Timedoctor
Coding Academy

8

MEMORY GAME: CHECK FOR MATCH



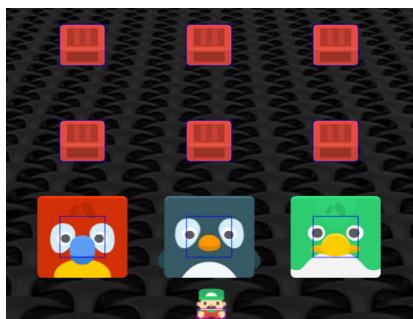
Apa Yang Akan Kita Pelajari?

1. Menggunakan Kelas Tweens
2. Mencocokkan Item



Memory Game

Pada pertemuan sebelumnya, kita sudah berhasil membuka Box atau memunculkan item Binatang.



- Tapi, apakah menurutmu bagus jika Item muncul tanpa animasi?
- Item juga menghalangi Box bukan?

Nah untuk membuat item dapat muncul lebih bagus, ayo kita **atur posisi munculnya Item dan juga Animasi** saat Item keluar dari Box. Kita akan menggunakan Tween dari **kelas Phaser.Tweens**.

TweenManager



Tween adalah animasi yang menggunakan key frame untuk menggerakkan objek dari satu titik ke titik lainnya.



Item Tween Animation

Method yang akan kita gunakan dari kelas Tween adalah **method add yang berguna untuk menambahkan animasi tween** yang baru pada **target** (objek yang ingin dianimasikan)

Method add memiliki banyak properties. Ini adalah beberapa properties yang akan kita gunakan.



Syntax!

```
this.tweens.add({
  target : //objek yg dianimasikan
  y: //titik y
  alpha: //opacity atau transparansi
  scale: //ukuran
  duration: //lama tween
  delay: //jeda waktu
  onComplete: //function yg dijalankan setelah tween
})
```

Sekarang, tambahkan kode ini di dalam method **openBox()** seperti pada tanda komentar ya!



Codes

```
box.setData('opened', true)
//tambahkan kode di bawah ini
item.setData('sorted', true)
item.setDepth(2000)

item.setActive(true)
item.setVisible(true)

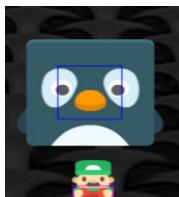
item.scale = 0
item.alpha = 0

this.tweens.add({
  targets: item,
  y: '--=50',
  alpha: 1,
  scale: 1,
  duration: 500,
})
```

Pertemuan 8 - Check for Match Tween Animation



Coba kamu lihat perubahannya



Before



After

Item Binatang sudah berada **lebih di atas**, tapi posisinya berada **di belakang Box**. Nah sekarang kita akan memindahkan Item Binatang agar berada di depan Box.

Selanjutnya tambahkan kode berikut ini di dalam method `update()` seperti pada tanda komentar

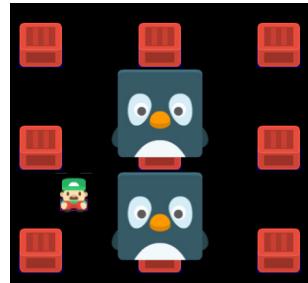


Codes

```
const child = c

//tambahkan kode di bawah ini
if (child.getData('sorted'))
{
    return
}
child.setDepth(child.y)
```

Lihat hasilnya pada game mu, pastikan Item sudah **tidak menghalangi Box** dan juga **memberikan Animasi Tween** saat keluar dari Box ya !





Tapi, coba kamu cek. Apakah Item muncul lagi serta beranimasi tween setiap spasi di klik?

Tentunya akan lebih bagus jika **Item tidak bisa muncul dan tween berkali-kali bukan?** Karena Box seharusnya hanya bisa dibuka satu kali saja.

Untuk itu, kita akan menambahkan kode berikut ini di dalam method `handlePlayerBoxCollide()` di baris pertama



Codes

```
handlePlayerBoxCollide() {  
    const opened = box.getData('opened')  
  
    if (opened)  
    {  
        return  
    }  
    //di bawah ini ada kode sebelumnya  
}
```

Kode yang baru ditambahkan itu berfungsi untuk mengecek, **jika box sudah terbuka (`opened = true`) maka return (nilainya tetap)**. Sehingga Item tidak akan Tween ulang lagi.

Nah sekarang coba lagi pada game mu dan lihat perubahannya ! Pastikan kamu hanya bisa membuka Box sekali saja ya !

Pertemuan 8 - Check for Match

Check for Match Item



Check for Match Item

Untuk mencocokkan Item, pertama kita harus tahu bagaimana mengatur agar Player hanya bisa membuka maksimal 2 box saja.

Kita memerlukan properti baru bertipe **array untuk menyimpan box dan item yang dipilih, diberi nama selectedBox**.

Inisialisasi properti selectedBox pada method init()



Codes

```
this.selectedBoxes = []
```

Lalu buat kode untuk menambahkan nilai array selectedBox pada method openBox(), sebelum kode tweens.add



Codes

```
this.selectedBoxes.push({ box, item })
```

Jadi, setelah keyboard spasi di klik, maka objek box dan item yang dipilih akan disimpan ke dalam array selectedBox lalu item akan beranimasi Tween keluar.

Selanjutnya tambahkan function on Complete pada tweens dibawah properti duration.



Codes

```
onComplete: () => {
  if (itemType === 0)
  {
    this.handleBearSelected()
    return
  }
```



Codes

```
if (this.selectedBoxes.length < 2) {  
    return  
}  
this.checkForMatch()  
}
```

function onComplete adalah function yang dijalankan setelah tweek selesai.

Pada onComplete, kode di bawah ini berfungsi mengecek jika itemType = 0 (**item binatang beruang**) maka panggil method **handleBearSelected()**

```
if (itemType === 0)  
{  
    this.handleBearSelected()  
    return  
}
```

Sedangkan kode di bawah ini berfungsi mengecek jika Box yang dipilih **kurang dari 2 (1 atau 0)** maka game tetap di jalankan (Player masih bisa membuka Box yang lain).

Tetapi, jika **pengecekan tidak dipenuhi (Box yang dibuka ada 2)** maka panggil method **checkForMatch()**

```
if (this.selectedBoxes.length < 2) {  
    return  
}  
this.checkForMatch()
```

Ini berarti, kita akan membuat **2 method** baru yaitu **handleBearSelected() dan checkForMatch()**

Pertemuan 8 - Check for Match

Check for Match Item



2. handleBearSelected()

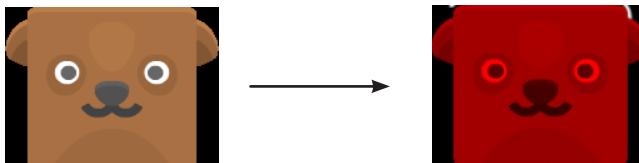
Method handleBearSelected() berfungsi untuk mengatur kode yang dijalankan saat **Player membuka Box yang berisi Item Beruang.**



Aturan pada Memory Game, Box yang berisi beruang tidak memiliki pasangan dan merupakan Box jebakan.

Untuk itu, ayo perhatikan gambaran umumnya dulu sebelum kita tuliskan ke dalam kode !

1. Beruang diatur warnanya menjadi merah dengan method setTint()



2. Saat player membuka Box berisi Item Beruang, player tidak bisa bergerak beberapa saat

3. Lalu tunggu 1 detik dan Item beruang tween masuk kembali

Nah setelah mengetahui apa yang akan terjadi saat method handleBearSelected() dipanggil, Ayo sekarang kita implementasikan ke dalam kode !



Buat method baru bernama handleBearSelected() dan tulis kode ini di dalamnya

Codes

```
handleBearSelected()
{
    const { box, item } = this.selectedBoxes.pop()

    item.setTint(0xff0000) //ubah warna beruang
    box.setFrame(20) //ubah frame box

    this.player.active = false //non active kan player
    this.player.setVelocity(0, 0) //tidak bisa bergerak

    this.time.delayedCall(1000, () => {
        item.setTint(0xffffffff)
        box.setFrame(7)
        box.setData('opened', false)

        this.tweens.add({
            targets: item,
            y: '+=50',
            alpha: 0,
            scale: 0,
            duration: 300,
            onComplete: () => {
                this.player.active = true
            }
        })
    })
}
```



Nah coba lihat hasilnya dan pastikan kodemu berhasil dijalankan ya !

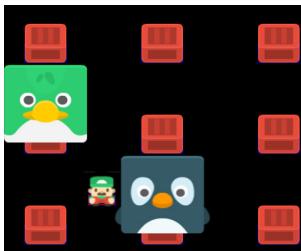


1. checkForMatch()

Selanjutnya adalah membuat method **checkForMatch()** untuk mengatur pencocokkan item.

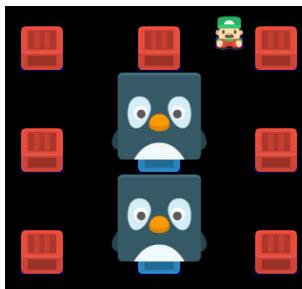
Perhatikan gambaran umum yang pertama dulu ya sebelum dituliskan ke dalam kode

1. Kondisi saat item yang muncul berbeda



Jika terpenuhi, Tween item masuk ke Box, dan lanjutkan game

2. Kondisi saat item yang muncul sama (kondisi no 1 tidak terpenuhi)



Maka biarkan Item tetap muncul dan ubah frame box



Nah dari gambaran tadi, ayo sekarang kita tuliskan ke dalam method checkForMatch()!

Buat method baru bernama checkForMatch() dan tulis kode ini di dalamnya

Codes

```
checkForMatch()
{
    const second = this.selectedBoxes.pop()
    const first = this.selectedBoxes.pop()

    if (first.item.texture !== second.item.texture)
    {
        this.tweens.add({
            targets: [first.item, second.item],
            y: '+=50',
            alpha: 0,
            scale: 0,
            duration: 300,
            delay: 1000,
            onComplete: () => {
                this.itemsGroup.killAndHide(first.item)
                this.itemsGroup.killAndHide(second.item)

                first.box.setData('opened', false)
                second.box.setData('opened', false)
            }
        })
    }
    return
}
```

Pertemuan 8 - Check for Match

Check for Match Item

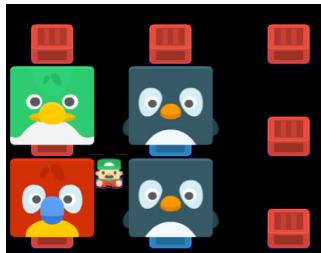


Codes

```
this.time.delayedCall(1000, () => {
    first.box.setFrame(8)
    second.box.setFrame(8)
})
}
```

Coba kamu buka Box yang berisi Item berbeda dan pastikan Item akan Tween masuk kembali!

Lalu coba juga membuka Box yang berisi Item sama!



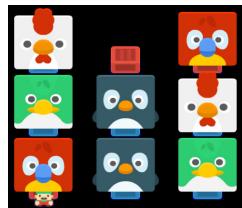
Nah selanjutnya, perhatikan gambaran umum yang kedua. Kita akan membuat kode untuk mengatur pencocokan item **saat semua item sudah berhasil dicocokkan**.

1. Simpan Hasil Pencocokan

Kita akan memerlukan properti baru diberi nama matchesCount bertipe number untuk menyimpan setiap Item yang berhasil dicocokkan.

2. Menambahkan kondisi baru Jika nilai matchesCount lebih dari 4

Karna kita memiliki 4 pasangan item, maka kita harus menambahkan perintah apa yang dilakukan saat semua item telah dicocokkan.



Nah dari gambaran tadi, ayo sekarang kita tambahkan ke dalam method checkForMatch()!



Pertama, inisialisasi dulu properti matchesCount di method init()

Codes

```
this.matchesCount = 0
```

Lalu tambahkan kode ini di dalam method checkForMatch() diantara penutup if dan kode delayedCall()

Codes

```
+this.matchesCount
```

Kode di atas berfungsi menambahkan nilai matchesCount setiap ada item yang cocok (item 1 == item 2).

Nah selanjutnya kita akan menambahkan kondisi sesuai dengan gambaran nomor 2.

Buat kode bertanda komentar di dalam delayedCall()

Codes

```
this.time.delayedCall(1000, () => {
    first.box.setFrame(8)
    second.box.setFrame(8)

    //tambahkan kode ini
    if (this.matchesCount >= 4)
    {
        this.player.active = false
        this.player.setVelocity(0, 0)
    }
})
```

Tambahkan kode ini di dalam method movePlayer paling atas.

Codes

```
if (!this.player.active) {
    return
}
```



Pertemuan 8 - Check for Match

Check for Match Item

Hari/Tanggal :

1. Apa itu tween? Bagaimana kode untuk menambahkan Tween Animation?

.....
.....
.....

2. Apa kegunaan properties target pada kelas tween?

.....
.....
.....

3. Apa fungsi properti selectedBox? Bagaimana inisialisasinya?

.....
.....
.....

4. Apa fungsi kode ini?

```
onComplete: () => {
    if (itemType === 0) {
        this.handleBearSelected()
    }
}
```

.....
.....
.....

5. Bagaimana kode untuk mendeklarasikan variabel spaceJustPressed ?

.....
.....
.....



Timedoctor
Coding Academy



Timedoctor
Coding Academy

q

MEMORY GAME: COUNTDOWN TIMER



Apa Yang Akan Kita Pelajari?

1. Menambahkan Timer
2. Menambahkan Game Lose
3. Menambahkan Game Win
4. Mendeploy Game



Memory Game



Warm Up!

Pada pertemuan sebelumnya, kamu sudah berhasil menyelesaikan inti game mu!

 Ayo coba kamu ingat-ingat lagi apa saja yang sudah ditambahkan dari pertemuan sebelumnya?

 Menurutmu apa yang akan kita lengkapi hari ini?



Seperti pada game lainnya, kita akan menambahkan kondisi **kalah dan menang** pada games !

Selain itu kita juga akan menambahkan **Timer** yang akan menghitung mundur



Ayo kita mulai dengan menambahkan Game Lose !



Game Lose and Timer

 Masih ingat tidak pada game Math Fighter kita juga menggunakan timer?

Nah konsep yang akan kita gunakan juga sama dengan Math Fighter loh! Jadi ayo ikuti petunjuk berikut ini dan buat kodennya secara mandiri ya !



Do it Your Self!

Kamu bisa menandai bagian yang sudah dikerjakan agar tidak bingung !

Inisialisasi properti untuk teks timer dan value timer

```
this.timerLabel = undefined  
this.countdownTimer = 40  
this.timedEvent = undefined
```

Create properti timerLabel dengan teks

```
this.add.text(this.halfWidth, 16, null)
```

Buat delay 1 detik lalu panggil method gameOver() secara berulang di method create()

```
this.timedEvent = this.time.addEvent({  
    delay: 1000,  
    callback: this.gameOver,  
    callbackScope: this,  
    loop: true  
})
```

Atur font style dan nilai dari teks Timer dengan properti countdownTimer di method update()

```
this.timerLabel.setStyle({  
    fontSize: '24px',  
    fill : '#ffffff',  
    fontStyle: 'bold',  
    align : 'center'  
}).setText(this.countdownTimer)
```

Pertemuan 9 – Countdown Timer Adding Game Lose



Selanjutnya, buat method gameOver()



Codes

```
gameOver ()  
{  
    this.countdownTimer -= 1  
    if (this.countdownTimer == 0) {  
        this.add.text(this.halfWidth, this.halfHeight +  
            250, 'You Lose!', {fontSize: 60})  
        .setOrigin(0.5)  
        this.countdownTimer = undefined  
        this.player.active = false  
        this.player.setVelocity(0, 0)  
    }  
}
```

Nah sekarang coba kamu uji game mu apakah sudah berhasil memunculkan teks You Lose saat waktu habis?



Game Win

Game menang jika Player **berhasil mencocokkan semua item Binatang sebelum waktu habis**. Jadi kita akan memunculkan teks "You Win" pada method checkForMatch()



Pergi ke `delayedCall()` di method `checkForMatch()` dan tambahkan kode ini seperti tanda komentar



Codes

```
if (this.matchesCount >= 4)
{
    this.player.active = false
    this.player.setVelocity(0, 0)

    //tambahkan kode di bawah ini
    this.add.text(this.halfWidth, this.halfHeight +
        250, 'You Win!', {fontSize: 60})
        .setOrigin(0.5)
    this.countdownTimer = undefined
}
```

Nah sekarang coba kamu menangkan game nya dan lihat apakah teks You Win sudah muncul?





Challenge!

Ayo buat game mu lebih bagus!

1. Ganti teks kalah dan menang dengan menambahkan scene baru untuk Win dan Game Over.
2. Desain layout scene mu menggunakan gambar-gambar yang dicari di Google.
3. Buat kode agar saat menang maka pindah ke Win Scene
4. Buat juga kode agar saat kalah maka pindah ke Game over Scene



Deploy to Netlify

Nah sekarang saatnya kita belajar Deploy Memory Game ke Netlify. Ikuti langkah berikut ini ya!

Ketik perintah berikut ini pada terminal

```
npm run build
```

Tunggu hingga proses selesai

```
dist\main.d11ea697.js.map    !! 7.8 MB      625ms
dist\main.d11ea697.js        955.27 KB    34.97s
dist\index.html              124 B       2.08s
```

Buka folder **dist** pada folder MemoryGame mu.

Lalu buka file **index.html**

Hapus tanda garis miring sebelum main dan simpan.

```
<script src="/main.1f19ae8e.js"></script>
```

Deploy folder dist tersebut pada Netlify. Ingat untuk mengganti alamat web nya ya !



Hari/Tanggal :

1. Properti apa yang digunakan untuk menyimpan nilai timer?

.....
.....
.....

2. Bagaimana kode untuk membuat teks pada Timer Label?

.....
.....
.....

3. Apa fungsi kode ini?

```
this.countdownTimer -= 1
```

.....
.....
.....

4. Apa yang akan terjadi saat kondisi berikut ini terpenuhi ?

```
if (this.countdownTimer == 0)
```

.....
.....
.....

5. Apa kondisi yang harus dipenuhi agar teks You Win muncul?

.....
.....
.....



Timedoctor
Coding Academy

10

MAKE YOUR OWN GAME: EXPLORING IDEA



Apa Yang Akan Kita Pelajari?

1. Memahami Proses Development
2. Mengeksplorasi Jenis Game yang Ingin Dibuat
3. Membuat Gambaran Umum Game



Make Your Own Game

Saatnya membuat Game dengan kreativitasmu sendiri !

Pada meeting sebelumnya kamu sudah membuat banyak game dari buku,
Sekarang adalah kesempatanmu untuk menuangkan ide mu sendiri ke dalam game !

Untuk melakukannya, kamu akan melalui proses development yang berisi langkah-langkah ini



1. Development Process



Warm Up!



Masih ingat apa saja langkah-langkah dalam membuat Game Project seperti pada Beginner 3?



1. Explore



2. Design



3. Coding



6. Deploy



5. Revise



4. Present



Nah pada meeting ini kita akan melakukan tahap yang pertama yaitu **Explore** !

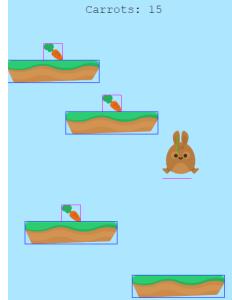
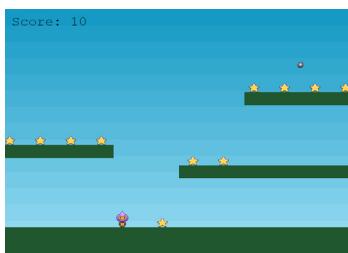


2. Choose Your Game Type

Lihat kembali game yang sebelumnya kamu buat dan jadikan referensi untuk game mu !

1. Platformer Game

Contoh game platformer adalah Collecting Star dan Bunny Jump karena menggunakan platform sebagai pijakan Player



2. Shooter Game

Contoh shooter game adalah Corona Buster yang memiliki Player yang bisa menembakkan Laser ke arah musuh.



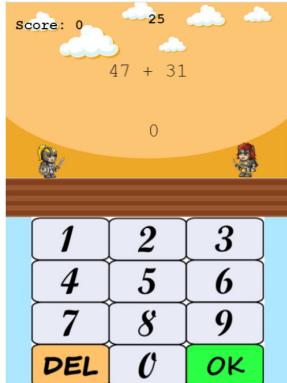
Pertemuan 10 - Make Your Own Game

Choose Game Type



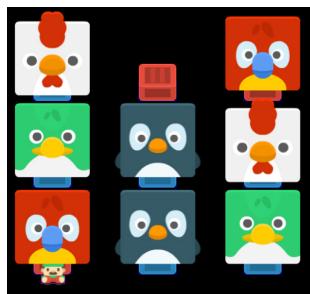
3. Science Game

Contoh science game adalah Math Fighter yang menggunakan sistem soal dan jawaban.



4. Memory Game

Contoh memory game adalah Memory Game yang menggunakan kemampuan mengingat untuk menyelesaikan game.



Sudah tahu ingin membuat game apa?



3. Make Your Own Game

Ayo kita rencanakan game yang akan kamu buat dengan mengisi form berikut ini ya!



1. Game Description

My Phaser Game

Judul Game :

Tipe Game :

Goal/Tujuan

.....
.....
.....
.....

Objek/Karakter Game

.....
.....
.....
.....

Kondisi Kalah

.....
.....
.....

Kondisi Menang

.....
.....
.....



2. Draft Design Game

Dari deskripsi yang sudah kamu tulis, **coba sekarang gambarkan draft desainnya ya !**

Jika kamu memiliki beberapa layout, kamu bisa membaginya dalam beberapa kotak.

Desain Game

 Kenapa kamu **harus mencatat deskripsi dan menggambar desain game?**

Agar kamu bisa belajar **merencanakan** sesuatu yang ingin kamu buat dan **membuat recordnya** sehingga tidak akan bingung saat ingin melanjutkan di lain hari!



Setelah mengetahui bagaimana game mu akan berjalan, saatnya untuk sampaikan kepada teman sekelasmu !



3. Exploring Game Assets

Game asset bisa kamu cari di Google loh! Ada banyak sekali Game Assets gratis yang bisa kamu dapatkan. Kamu juga bisa menggambarnya sendiri.

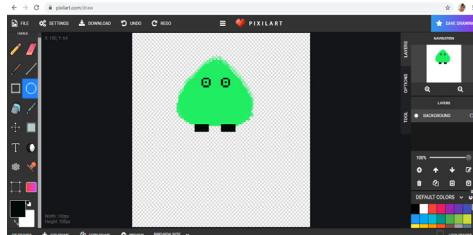
1. Cari Game Assets gratis di link ini :

- <https://itch.io/game-assets/free>
- <https://www.kenney.nl/assets>
- <https://www.gamedevmarket.net/>
- <https://www.gameart2d.com/freebies.html>
- <https://craftpix.net/freebies/>



2. Gambar sprite mu sendiri di link ini :

- <https://www.piskelapp.com/>
- <https://makepixelart.com/>
- <https://www.pixilart.com/draw>





3. Cari audio atau sound effect di link ini :

<https://soundible.com/free-sound-effects-1.html>

<https://orangefreesounds.com/>

<https://opengameart.org/content/library-of-game-sounds>



3. Conclusion

Terakhir, ayo kita buat target penggeraan game mu selama meeting 10 sampai meeting 15.

Sejauh mana kamu ingin menyelesaikannya dalam 1 meeting?

| Meeting | Target |
|---------|--------|
| 10 | |
| 11 | |
| 12 | |
| 13 | |
| 14 | |
| 15 | |



Apa saja yang kamu kerjakan hari ini?

.....
.....
.....
.....

Cata di website apa kamu mendapatkan asset yang menarik!

.....
.....
.....
.....



Timedoctor
Coding Academy

11

MAKE YOUR OWN GAME: START DESIGN



Apa Yang Akan Kita Pelajari?

1. Mengetahui Contoh Aplikasi Populer
2. Menyiapkan Template
3. Mendesain Scene



Make Your Own Game



**Sudah sejauh mana kemampuan coding mu?
Apa kamu punya mimpi untuk membuat Aplikasi besar
dan keren yang akan digunakan oleh banyak orang?**



1. Popular Local Apps

Di Indonesia, ada banyak programmer sukses yang sudah membuat aplikasi keren loh ! Bahkan aplikasinya juga digunakan oleh negara lain.



GOJEK

Siapa sih yang tidak tahu GOJEK ?

GOJEK adalah aplikasi buatan PT Aplikasi Karya Anak Bangsa yang menyediakan layanan pesan antar dan layanan lainnya untuk masyarakat umum.

Yuk cek video ini!



GOJEK. A Super App.

<https://www.youtube.com/watch?v=Tn4MGnTkF8c>



TRAVELOKA adalah aplikasi milik PT Traveloka Indonesia yang menyediakan jasa travel dan segala keperluannya.

Yuk cek video ini!



Introduction to Traveloka App

<https://www.youtube.com/watch?v=YltFGWVWiGo>



GAME TAHU BULAT

TAHU BULAT adalah game buatan kreator Own Games yang memiliki challenge menjadi penjual tahu bulat .

Yuk cek video ini!



<https://www.youtube.com/watch?v=WhUmJpNnMEc>



Nah. Bagaimana menurutmu tentang 3 video tadi?
Apa kamu punya ide untuk membuat aplikasi serupa?

Yuk kita mulai dari game sederhana yang akan kamu buat sendiri di Intermediate 3 ini !

“It does not matter how slowly you go as long as you do not stop” -Confucius



2. Start Design

Sebelumnya kamu sudah mencari game assets yang akan digunakan dalam game mu.

Nah, hari ini kita akan melanjutkan dengan **persiapan template dan desain scene atau layout game.**



1. Persiapan Template

Seperti biasa lakukan persiapan template phaser di folder Phaser mu.

- Copy Paste Template Phaser
- Masukkan Game Assets ke folder public game mu
- Buat Game Scene dan Strukturnya
- Konfigurasi main.js
- Edit index.html

Coba jalankan **npm run start** dan pastikan phaser berhasil muncul di browser mu



2. Desain Scene

Desain scene atau layout game sesukamu, mulai dari **menambahkan image, sprite sheet, dll.** Pastikan penempatan image mu tepat dan rapi ya !



Code Hint

1. Kode untuk mengatur scene ada di tengah-tengah halaman browser

Tulis pada main.js



Codes

```
scale: {  
    mode: Phaser.Scale.FIT,  
    autoCenter: Phaser.Scale.CENTER_BOTH  
}
```

- 2.

Sintaks load image, sprite sheet, audio

Tulis pada method preload()



Codes

```
this.load.image('keyName', 'location')  
this.load.spritesheet('keyName', 'location',  
{frameWidth: , frameheight: })  
this.load.audio('keyName', 'location')
```

frameWidth dan frameHeight diisi dengan ukuran framenya



3. Sintaks create image, sprite, static group, group

Tulis pada method create()



Codes

```
this.add.image(x, y, 'keyName')  
this.physics.add.sprite(x, y, 'keyName')  
this.physics.add.staticGroup()  
this.physics.add.group()
```



3. Conclusion

Apa saja yang kamu kerjakan hari ini?

.....
.....
.....
.....

Image apa yang kamu gunakan?

.....
.....
.....
.....

Background apa yang kamu gunakan?

.....
.....
.....
.....



Sound Effect apa yang kamu pilih?

.....
.....
.....
.....

Apa target mu pada meeting ini tercapai?





Timedoctor
Coding Academy

12

MAKE YOUR OWN GAME: CODE THE LOGIC



Apa Yang Akan Kita Pelajari?

1. Menggerakkan Karakter Game
2. Membuat Karakter Game
3. Membuat Kelas dan Method
4. Mengatur Collider dan Overlaps



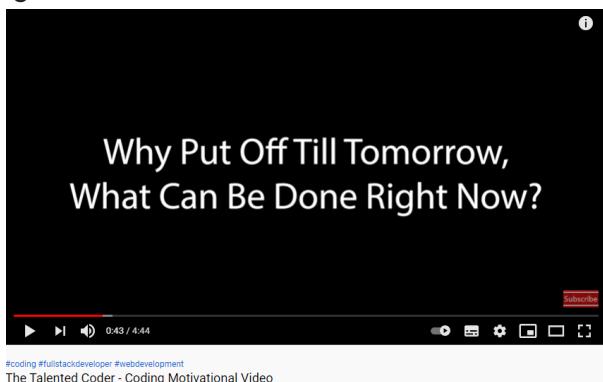
Make Your Own Game



Apa motivasi kamu saat belajar atau ketika menemukan kesulitan dalam menyelesaikan masalah?
Apakah kamu punya Quotes Favorit?

Setiap hari pasti kamu akan menemukan masalah. **Nah sama halnya dengan coding, kamu pasti pernah menemukan error.**
Itu adalah hal yang wajar.

Yuk cek video ini agar kamu lebih semangat mengerjakan game mu !



#coding #fullstackdeveloper #webdevelopment
The Talented Coder - Coding Motivational Video

<https://www.youtube.com/watch?v=fxlIVbvOHY&t=85s>

Bagaimana dengan video tadi?
Apa yang bisa kamu pelajari dari video itu?



1. Code The Logic

Kali ini kita akan mulai untuk mengerjakan logika game seperti bagaimana **karakter bergerak dan beranimasi**, apa saja **kelas dan method yang diperlukan**, karakter apa yang **tidak bisa ditembus**, dan lain-lain



Code Hint

1. Contoh kode untuk menggerakkan karakter dengan key arrow ada 2 cara.

Tambahkan dulu kode input cursor

```
this.input.keyboard.createCursorKeys()
```

Cara 1: dengan setVelocity

Tulis pada method update() atau buat method baru lalu panggil di method update()



Codes

```
if (this.cursors.left.isDown)
{
    this.player.setVelocity(-70, 0)
}
```

velocity :



-70, 0



70, 0



0, 70



0, -70



Cara 2 : dengan setVelocityX dan setVelocityY

Tulis pada method update() atau buat method baru lalu panggil di method update()



Codes

```
if (this.cursors.right.isDown)
{
    this.player.setVelocityX(70)
}
```

velocity X :



negatif



positif

velocity Y :



negatif



positif

Kamu juga bisa mengganti angka 70 itu dengan membuat variabel speed.

2.

Contoh kode untuk animasi karakter

Animasi karakter ini dapat digunakan pada asset berbentuk sprite sheet.



Ingin saat menghitung jumlah frame dimulai dari 0 ya!

Tulis kode pada method create() atau buat method baru lalu panggil di method move player.



Codes

```
this.anims.create({  
    key: 'keyName', //contohnya 'left'  
    frames: this.anims.generateFrameNumbers('playerKey',  
        {start: , end: }),  
    frameRate: 10,  
    repeat: -1  
})
```

start → dimulai dari nomor frame berapa

end → berakhir di nomor frame berapa

Buat untuk semua animasi !

3. Contoh kode untuk kelas Objek yang bergerak (Enemy, Laser, Item)

Pada umumnya, kelas objek berisi method untuk **membangun kelas, memunculkan objek, menghapus objek, dan update.**

Membangun kelas (constructor)

Constructor umumnya berisi parameter **scene, x, y, texture, config**. Consturctor bisa digunakan untuk menginisialisasi properti (speed, rotation, dll) dan juga mengatur ukuran objek.

Tulis kode ini di kelas baru



Codes

```
constructor(scene, x, y, texture, config) {  
    super(scene, x, y, texture)  
    this.setScale(2)  
    this.speed = 200  
    this.rotation = config.rotation  
}
```

Pertemuan 12 – Make Your Own Game

Code The Logic



Memunculkan Objek (spawn)

Pada method spawn, panggil method Phaser berikut ini



Codes

```
spawn(x, y) {  
    setPosition(x, y)  
    setActive(true)  
    setVisible(true)  
}
```

Nilai x dan y pada setPosition juga bisa memanfaatkan **Phaser.Math.Between()** untuk mencari angka acak

Menghapus Objek (delete)

Pada method delete, panggil method Phaser berikut ini



Codes

```
delete() {  
    this.destroy()  
}
```

Method update()

Pada dasarnya method update bisa menggunakan kode-kode ini untuk menggerakkan objek.

- menggerakkan ke atas



Codes

```
update(time) {  
    this.setVelocityY(this.speed)  
}
```



- menggerakkan ke bawah



Codes

```
update(time) {  
    this.setVelocityY(this.speed) * -1  
}
```

- menggerakkan ke kanan



Codes

```
update(time) {  
    this.setVelocityX(this.speed)  
}
```

- menggerakkan ke kiri



Codes

```
update(time) {  
    this.setVelocityX(this.speed) * -1  
}
```

Selain itu, di method update() juga bisa membuat kondisi kapan objek itu akan menghilang. Contohnya :



Codes

```
if(this.y < -10) {  
    this.erase()  
}
```

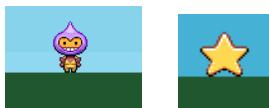
Pertemuan 12 – Make Your Own Game Code The Logic



4. Contoh kode untuk Collides dan Overlaps

Collides dan Overlap sama-sama bisa digunakan untuk membuat objek 1 tidak bisa menembus objek 2 dan sebaliknya.

Pada umumnya, **Collides** digunakan untuk membuat Karakter berdiri di atas Platform, Bola yang memantul di atas Paddle, dll



Sedangkan **Overlap** digunakan ketika Karakter mengumpulkan Coin atau Laser menyentuh Enemy.

Sintaks Collider sama dengan Overlap



Codes

```
this.physics.add.collider(object1, object2,  
collideCallback, processCallback, callbackContext)  
  
this.physics.add.overlap(object1, object2,  
collideCallback, processCallback, callbackContext)
```

Contoh kode:

Tulis kode Collider atau Overlap pada method create()



Codes

```
this.physics.add.collider(this.player, this.  
platform)  
  
this.physics.add.collider(this.player, this.coin,  
this.collectCoin, null, this)
```



5. Contoh kode untuk Check Collision

Check Collision digunakan untuk mengatur sisi mana yang bisa menembus dari sebuah objek.

Tulis kode check collision di method create()

Contoh kode :



Codes

```
this.player.body.checkCollision.up = false // (tembus)
```

Kode yang sama juga bisa digunakan untuk **down, left, dan right.**



3. Conclusion

Apa saja yang kamu kerjakan hari ini?

.....
.....
.....
.....

Apa target mu pada meeting ini tercapai?





Timedoctor
Coding Academy

13

MAKE YOUR OWN GAME: LET'S CONTINUE!



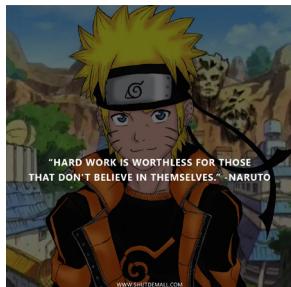
Apa Yang Akan Kita Pelajari?

1. Melanjutkan Game Project
2. Menambahkan Label
3. Menambahkan Timer Event



Make Your Own Game

Ayo lanjutkan Game mu !



1. Let's Continue



Code Hint

1.

Contoh kode untuk membuat Label

Dalam membuat label (teks) bisa menggunakan 2 cara.

Cara 1: Tanpa Kelas

Menambahkan Label tanpa kelas artinya membuat kodennya di kelas Game Scene.

Pertama, tambahkan dulu teksnya di method `create()`.
Contoh kode :



Codes

```
this.scoreText = this.add.text(240, 10,  
'Score: 0', { color: '#000', fontSize: 24 })  
.setScrollFactor(0)
```



Selanjutnya kamu perlu membuat properti baru untuk menyimpan nilai Score yang bisa berkurang atau bertambah

Buat dulu properti baru di method init()



Codes

```
this.score = 0
```

Pada method yang mengatur kondisi kapan Score akan bertambah (misalnya collectCoin()), buat kode untuk mengubah Teks menjadi nilai Score.

Contoh kode:



Codes

```
this.score++  
this.scoreText.text = `Score: ${this.score}`
```

Cara 2 : Dengan Kelas

Menambahkan Label dengan kelas artinya membuat kelas baru yang mengatur objek label dan menggunakannya pada Game Scene.

Pertama, buat kelas baru dengan menambahkan file JavaScript baru di dalam folder src. Kamu bisa membuatkan folder baru agar lebih rapi.

Pada umumnya, kelas Label berisi method untuk membangun kelas (**constructor**), mengatur nilai (**set**), mengambil nilai (**get**), dan **menambah/mengurangi nilai**.



Contoh kode:

Codes

```
import Phaser from 'phaser'

const formatScore = (gameScore) => `Score: ${gameScore}`

export default class ScoreLabel extends Phaser.GameObjects.Text
{
    constructor(scene, x, y, skor, style)
    {
        super(scene, x, y, formatScore(skor), style)
        this.score = skor
    }

    setScore(skor)
    {
        this.score = skor
        this.setText(formatScore(this.score))
    }

    getScore()
    {
        return this.score
    }

    add(points)
    {
        this.setScore(this.score + points)
    }
}
```



Selanjutnya adalah membuat method baru di Game Scene dimana kelas Label akan dipanggil.

Import dulu kelas ScoreLabel pada Game Scene

Lalu buat method baru dengan nama bebas sesukamu. Misalnya bernama createScoreLabel(). Contoh kode :



```
createScoreLabel(x, y, score)
{
    const style = { fontSize: '24px', fill: '#000',
        fontStyle: 'bold' }
    const label = new ScoreLabel(this, x, y, score,
        style).setDepth(1)

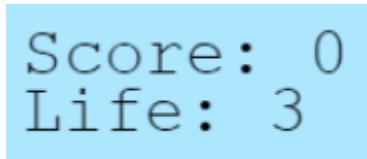
    this.add.existing(label)
    return label
}
```

Sesuaikan nama kelas, text style, dan depth dengan game mu.

Selanjutnya **inisialisasi this.scoreLabel = 0 di method init()**

Lalu **panggil method createScoreLabel() di method create()**.

```
this.scoreLabel = this.createScoreLabel(26, 16, 0)
```



Score: 0
Life: 3



2. Contoh kode untuk membuat Timer Event

Timer Event digunakan untuk memberi jarak waktu sebelum memanggil method selanjutnya.

Sintaks Timer Event



Codes

```
this.time.addEvent({  
    delay: ,  
    callback: ,  
    callbackScope: ,  
    loop:  
})
```

Timer Event biasanya dipanggil pada method `create()`

Contoh Kode



Codes

```
this.time.addEvent({  
    delay: 2000,  
    callback: this.spawnEnemy,  
    callbackScope: this,  
    loop: true  
})
```

Artinya setiap memanggil method `spawnEnemy`, tunggu (delay) selama 2 detik.



2. Let's Play Games

Ayo kita peregangan sebentar dengan bermain game !
Gamenya bernama **Depend on Your Name.**



Aturannya adalah membuat sebuah kalimat berdasarkan huruf depan nama lengkapmu !

Contoh :

Nama lengkap : **Mikasa Ackerman**
Kalimat games : **Mendarat di Angkasa**

Kalimat tidak boleh sembarang dan kata depan dan penghubung seperti di, dan, ke, dll tidak termasuk ya !
Kamu bisa menentukan **hukuman** untuk yang terakhir menyusun kalimat !



3. Conclusion

Apa saja yang kamu kerjakan hari ini?

.....
.....
.....
.....

Apa target mu pada meeting ini tercapai?





Timedoctor
Coding Academy

14

MAKE YOUR OWN GAME: DEBUG AND REVISE



Apa Yang Akan Kita Pelajari?

1. Melanjutkan Game Project
2. Membuat Start Game dan Game Over
3. Melakukan Debugging
4. Mempresentasikan Progress



Make Your Own Game



1. How to Debug?

Selama membuat Game Project dari Meeting 10 sampai hari ini, kamu selalu melakukan debugging bukan?

Debugging atau proses mencari dan memperbaiki kesalahan program adalah bagian dari programming.



Bagaimana programmer melakukan debugging?

Yuk Cek Video Ini!



YouTrack - The Art of Bug Tracking

Nah jadi jangan takut untuk melakukan kesalahan saat melakukan programming ya, **pasti selalu ada solusi!**



2. Code The Logic



Code Hint

1.

Contoh kode untuk menambahkan Game Start

Dalam membuat Game Start kamu bisa membuat scene khusus untuk layout Start.

Buat scene baru dengan nama sesukamu. Misalnya bernama **StartScene**.

Lalu desain Star Scene mu.

Untuk berpindah ke Game Scene kamu bisa memanfaatkan **Button** atau dengan **Jeda Waktu**.

Button

Tambahkan sebuah image button

Lalu tambahkan kode **setInteractive()** pada kode menambahkan image button.

Contoh kode :



Codes

```
this.startButton = this.add.image(200, 500,  
'start-button').setInteractive()
```

Lalu tambahkan kode berikut ini di bawahnya untuk berpindah ke Game Scene.

Pertemuan 14 – Make Your Own Game

Debugging



Codes

```
this.startButton.once('pointerup', () => { this.scene.start('game-scene') }, this)
```

Sesuaikan dengan nama start button dan key game scene mu ya!

Jeda Waktu

Jika menggunakan jeda waktu, kamu tidak memerlukan button. Setelah mendesain layout Start Scene, **tambahkan kode ini pada method create()**



Codes

```
this.time.addEvent({  
    delay : 3000,  
    callback : this.startGame,  
    callbackScope : this,  
    loop : true  
})
```

Lalu tambahkan method baru bernama **startGame()**



Codes

```
startGame()  
{  
    this.scene.start('corona-buster-scene')  
}
```



2. Contoh kode untuk menambahkan Game Over

Buat dulu scene baru dengan nama sesukamu, contohnya bernama GameOverScene.

Lalu desain layout Game Over mu.

Jika ingin **menampilkan Score**, ikuti kode berikut ini.

Pertama, tambahkan method init dengan parameter data.



Codes

```
init(data)
{
    this.score = data.score
}
```

Lalu tambahkan kode ini di method create()



Codes

```
this.add.text(70, 300, 'SCORE:' + this.score,
{ fontSize: '60px', fill: '#000' })
```

Jika ingin **menambahkan button Replay**, caranya sama dengan menambahkan button start ya !

Nah selanjutnya kamu bisa tambahkan kode untuk berpindah ke game over scene pada kondisi kalah. Contohnya saat life = 0, pindah ke game over scene



Codes

```
if (this.lifeLabel.getLife() == 0) {
    this.scene.start('game-over-scene',
        { score: this.scoreLabel.getScore() })
}
```



3. Brainstorming

Nah sekarang saatnya kamu memperlihatkan progress project mu ke teman-teman mu !

Saat memperlihatkannya, lakukan beberapa hal ini ya !

1. Jelaskan Nama Game mu dan Deskripsinya

Cobalah untuk menjelaskan game mu sambil memainkannya.



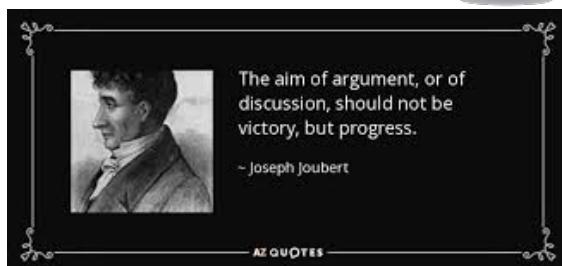
2. Tanyakan pada teman jika kamu memiliki permasalahan

Jangan malu untuk bertanya dalam kelompok belajar ya!



3. Minta pendapat teman tentang game mu

Tanya pada temanmu jika mereka punya pendapat untuk improve game mu



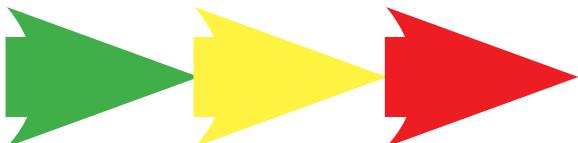


4. Revise



Saatnya mengimplementasikan saran temanmu!
Apakah semua saran ingin kamu buat?

Kamu bisa mulai dari saran yang paling mudah ya!



5. Conclusion

Apa saja yang kamu kerjakan hari ini?

.....
.....
.....
.....

Apa target mu pada meeting ini tercapai?





Timedoctor
Coding Academy

15

MAKE YOUR OWN GAME: FINISHING AND TESTING



Apa Yang Akan Kita Pelajari?

1. Melanjutkan Game Project
2. Melakukan Finishing dan Testing
3. Mempersiapkan Presentasi



Make Your Own Game



1. How to Be a Software Tester



Bagaimana dengan game mu? Apakah sudah puas dengan hasilnya?

Nah sekarang saatnya melakukan **testing** untuk memeriksa apakah semua kode sudah berjalan dengan baik dan sesuai!

Yuk Cek Video Ini!



#KUSKIS #AnimatedShortFilms
CG short film on God testing a software | "Tales from the Multiverse" - by Tumblehead

Apa yang bisa kamu pelajari dari video di atas?

Nah kamu juga harus melakukan testing pada project mu ya !
Pastikan semua pergerakan di game sesuai dengan apa yang diharapkan.



2. Things You Can Improve Next Time

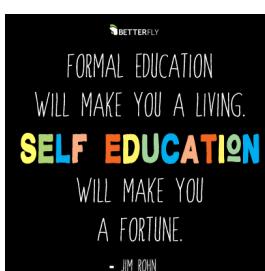
Setelah melakukan testing,
Apakah menurutmu ada yang bisa kamu **tingkatkan** dari game mu suatu saat nanti?

Misalnya **menambahkan lebih banyak level, membuat desain layout lebih original, menambahkan variasi tingkat kesulitan, dan yang lainnya.**

Yuk coba tuliskan idemu di bawah ini, apa yang ingin kamu tingkatkan dari game mu!

| No. | Bagian yang ingin di ubah/tingkatkan |
|-----|--------------------------------------|
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

Kamu tidak harus menambahkannya sekarang, saat punya waktu luang dan ingin meningkatkan kemampuanmu secara mandiri kamu bisa melanjatkannya ya!





3. Prepare Your Presentation

Pada meeting berikutnya, kamu akan mempresentasikan hasil game mu loh !

Ayo kita buat sama-sama presentasi mu di kelas ya. Ikuti langkah-langkah berikut ini!

1. Siapkan presentasi untuk 3-5 menit.



2. Kamu bisa membuatnya di Microsoft Power Point, Canva, atau yang lainnya. Tambahkan gambar kesukaanmu.

3. Presentasi harus berisi poin ini:



Pembuka

- Nama, Kelas, dan Salam Pembuka



Ide

- Goal/Tujuan Game
- Cara bermain
- Kondisi kalah dan menang



Desain

- Assets yang digunakan (image, sprite sheet, sound effect)



Kode

- Method yang digunakan dan fungsinya
- Kelas apa saja yang dibuat



Saran yang diterima

- Sebutkan saran yang kamu terima dari teman



Penutup

- Tambahkan kesan mu selama belajar di Timedoors
- Quotes (opsional)
- Salam Penutup



4. Conclusion

Apa saja yang kamu kerjakan hari ini?

.....
.....
.....
.....

Apa target mu pada meeting ini tercapai?





Timedoctor
Coding Academy

16

MAKE YOUR OWN GAME: DEPLOY AND SHARE



Apa Yang Akan Kita Pelajari?

1. Presentasi Project
2. Deploy ke Netlify
3. Bagikan Linknya ke Temanmu



Make Your Own Game



1. Presentation

Saatnya mempresentasikan hasil projectmu !

Bergiliran dengan temanmu, **presentasikan yang sudah kamu buat di pertemuan sebelumnya dan jangan lupa minta temanmu berkomentar ya!**



2. Deploy to Netlify



Masih ingat caranya deploy ke Netlify?

Ketik perintah berikut ini pada terminal

```
npm run build
```

Tunggu hingga proses selesai

```
dist\main.d11ea697.js.map    !! 7.8 MB      625ms
dist\main.d11ea697.js        955.27 KB     34.97s
dist\index.html               124 B        2.08s
```



Buka folder **dist** pada folder Game mu.

Lalu buka file **index.html**

Hapus tanda garis miring sebelum main dan simpan.

```
<script src="/main.1f19ae8e.js"></script>
```

Deploy **folder dist** tersebut pada Netlify. Ingat untuk **mengganti alamat web nya ya !**

SHARE

Nah sekarang **bagikan link game** mu ke teman sekelas agar temanmu bisa saling mencoba



Ayo **sampaikan pendapatmu** tentang game-game yang temanmu buat !

Ingat untuk menggunakan kata-kata yang sopan dan tidak menyakiti perasaan teman mu ya !



4. Conclusion

Apa saja pelajaran yang dapat kamu ambil dari pertemuan hari ini?

.....
.....
.....
.....

Bagaimana perasaanmu setelah Game mu selesai ?

