MODUL 4 GAME BERBASIS TEXT DAN GAME 2D COIN DASH

Tujuan:

1. Mahasiswa memahami variable local, global dan penggunaan onready
2. Mahasiswa membuat program yang menggunakan array dan pengkondisian if
3. Mahasiswa mampu menggunakan method append dan variable size
4. Mahasiswa mampu mengatur pergerakan Player pada Game Coin Dash

A. Variabel Lokal dan Global

Sama hal nya seperti pada Bahasa pemrograman lainnya, bahwa ketika mendefinisikan variable, terdapat variable local dan global. Seperti pada Program 4.1

Program 4.1

```
1  extends Control
2  var player_words = []
3
4  func _ready():
5      var prompts = ["Jerry" , "Apel basi" , "Mual" , "satu"]
6      var story = "Pada suatu hari %s memakan %s dan ia merasa %s selama %s minggu"
7      print (story % prompts)
8      $VBoxContainer/DisplayText.text = story % prompts
9
10 func _on_PlayerText_text_entered(new_text):
11     update_DisplayText(new_text)
12     story = "
```

Pada line 2, terdapat variable global player_words, variable dapat diakses di function manapun. Sementara pada line 6 terdapat variable local, yaitu variable story. Karena variable story hanya berlaku pada area local di function _ready( ), maka tidak dapat dipanggil di line 12 dan akan menyebabkan error. Bagaimana agar variable story dapat diakses di function _on_PlayerText_text_entered ?

Tugas 4.1

1. Mari perbaiki Program LoonyLips.gd
   Buatlah agar variable prompts dan variable story menjadi variable global
2. Tambahkan sebuah variable bernama PlayerText, dimana akan berisi node $VBoxContainer/HBoxContainer/PlayerText , sehingga lebih baik lagi dalam pengetikan program, dan mempermudah proses pemanggilan $VBoxContainer/HBoxContainer/PlayerText pada function-function lainnya.
3. Pada deklarasi variable
   ```
   var PlayerText = $VBoxContainer/HBoxContainer/PlayerText
   ```
   Tambahkan keyword **onready**
   Kata kunci onready dapat digunakan untuk menunda inisialisasi variabel hingga node menjadi bagian dari scene tree. Ini sangat membantu jika Anda perlu mengakses node lain dari scene tree untuk inisialisasi.
   ```
   onready var PlayerText = $VBoxContainer/HBoxContainer/PlayerText
   ```
4. Buat variable global DisplayText yang berisi $VBoxContainer/DisplayText. Sama seperti pada langkah 3
5. Perbaiki keseluruhan program LoonyLips.gd

6. Mari kita jadikan function _ready menjadi blok yang dikomentari. Caranya, blok seluruh function, lalu tekan Ctrl + k

B. Penggunaan append pada Array

Mari lanjutkan kembali Tugas 4.1 dengan membuat function baru. Function ini bertujuan untuk mengambil kata-kata dari player, dan akan dilakukan proses append ke dalam array

7. Tambahkan function baru add_to_player_words( ):
   Function baru ini akan mengambil kata-kata dari player dan menambahkan kata-kata tersebut ke dalam array menggunakan append. Silahkan cari function append di dokumentasi godot
8. Lengkapi isi dari function menjadi seperti pada Program 4.2

Program 4.2

```
1  extends Control
2  var player_words = []
3  var story = "Pada suatu hari %s memakan %s dan ia merasa %s selama %s minggu"
4  var prompts = ["Jerry" , "Apel basi" , "Mual" , "satu"]
5
6  onready var PlayerText = $VBoxContainer/HBoxContainer/PlayerText
7  onready var DisplayText = $VBoxContainer/DisplayText
8
9  #func _ready():
10 #    print (story % prompts)
11 #    DisplayText.text = story % prompts
12
13 func _on_PlayerText_text_entered(new_text):
14     pass
15
16 func _on_TextureButton_pressed():
17     pass
18 func add_to_player_words():
19     player_words.append(PlayerText.text)
```

9. Selanjutnya, ketika _on_PlayerText_text_entered( ), akan memanggil add_to_player_words dengan tujuan setiap apa yang diketik oleh player akan di-append ke dalam array player_words
10. Namun proses pengisian akan terus berlangsung. Sehingga buatlah sebuah function untuk memeriksa apakah player_words sudah selesai diisi atau belum? Dan function akan mengembalikan ukuran dari array tersebut apakah sudah sesuai dengan ukuran yang diinginkan. Jika sama akan mengembalikan True. Disini untuk memeriksa apakah ukuran dari kata-kata yang dimasukkan oleh Player akan sama dengan prompts? Karena nantinya isian dari prompts dimasukkan dari player
    Buatlah function is_story_done( )

```
func is_story_done():
    return player_words.size() == prompts.size()
```

Carilah di dokumentasi mengenai segala macam jenis instance variable sebuah objek array

C. Menggunakan elif

Di dalam GDScript terdapat pengkondisian if dan else if. Adapun beberapa pengkondisian seperti berikut ini:

1) If
2) Elif
3) Else

Perhatikan Potongan program berikut ini:

```
24 func check_player_words_length():
25     if is_story_done() :
26         tell_story()
27     if is_story_done():
28         pass
29     else :
30         pass
```

Jika pada line 25 menghasilkan nilai true, maka line 26 akan dikerjakan. Namun jika pada line 25 menghasilkan nilai false, line 26 tidak dikerjakan. Namun terlepas dari hasil apapun di line 25, berikutnya program akan membaca line 27. Artinya itu pengkondisian tersebut tidak ada kaitannya. Line 25 dan line 27 akan tetap dibaca. Namun jika anda bermaksud menjadi satu kesatuan blok seperti if else if, maka perhatikan potongan program berikut ini:

```
32 func check_player_words_length():
33     if is_story_done() :
34         tell_story()
35     elif not is_story_done():
36         pass
37     else :
38         pass
```

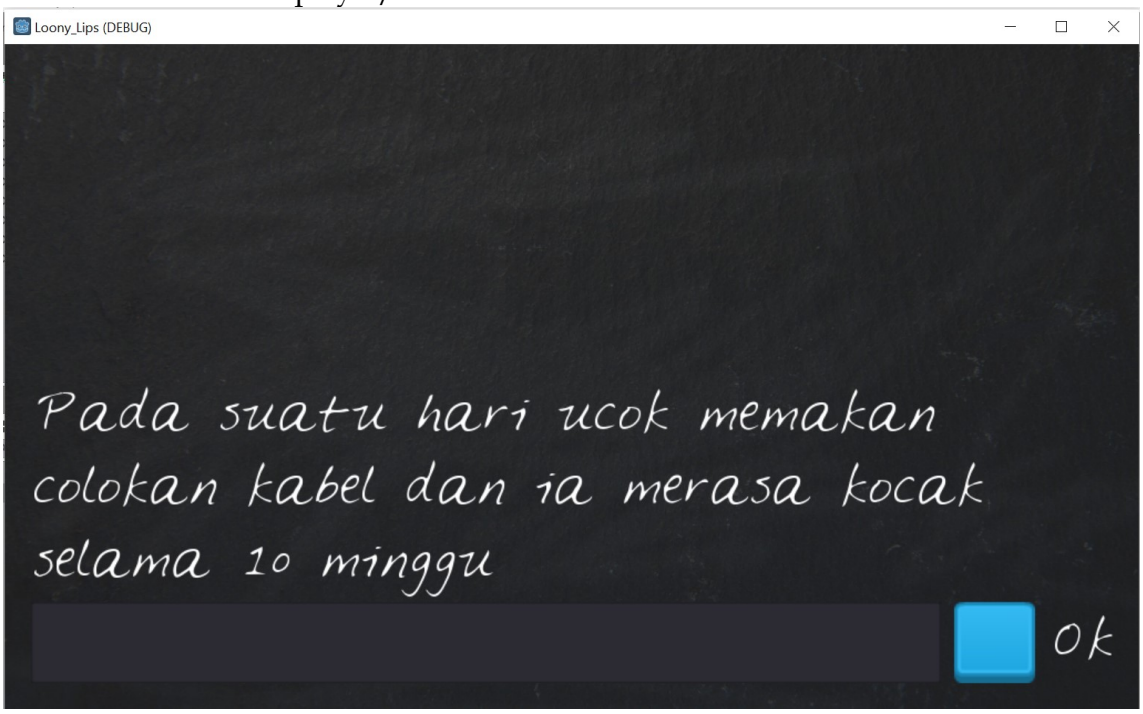
Jika line 33 menghasilkan true, maka akan mengerjakan line 34 dan tidak mengerjakan line 35. Namun jika line 33 menghasilkan false, kemudian akan masuk ke blok pengkondisian berikutnya yaitu pada line 35. Jika line 35 menghasilkan true, akan mengerjakan line 36, jika menghasilkan false akan mengerjakan line 37.

D. Game Mechanics

Mechanics dari game :

1. Terdapat **story** yang menyimpan isi cerita, dimana beberapa variable di dalam cerita tersebut diisikan oleh player/user. Masukan dari player/user akan ditampung di dalam array **player_words**
2. Ketentuan jenis variabelnya disimpan di dalam array **prompts**

3. Terdapat array **player_words** yang akan menampung masukan dari player, yang ditentukan harus sesuai dengan permintaan dari array **prompts**. Artinya Panjang array **player_words** harus sama dengan panjang array **prompts**.
4. Akan ditampilkan permintaan kepada player/user untuk memasukkan nilai kepada **player_words**. Setiap masukan dari player akan ditambahkan ke dalam array **player_words**
5. Setiap user/player selesai memasukkan satu buah kata, layar akan dibersihkan
6. Terdapat mekanisme pemeriksaan apakah player telah memasukkan jumlah kata sesuai panjang array **prompts**
7. Jika panjang array **player_words** sudah sama dengan panjang array **prompts**, maka artinya proses memasukan kata sudah selesai, dan story siap ditampilkan
8. Story akan terlihat seperti pada Gambar 4.1, dimana nilai-nilai prompts dimasukkan oleh player/user



Gambar 4.1

Sehingga untuk mendukung mechanics tersebut, script akan menjadi seperti pada **Program 4.3**

```
1  extends Control
2  var player_words = []
3  var story = "Pada suatu hari %s memakan %s dan ia merasa %s selama %s minggu"
4  var prompts = ["Nama" , "Kata Benda" , "Kata Keterangan" , "Angka"]
5  onready var PlayerText = $VBoxContainer/HBoxContainer/PlayerText
6  onready var DisplayText = $VBoxContainer/DisplayText
7
8  func _ready():
9      check_player_words_length()
```

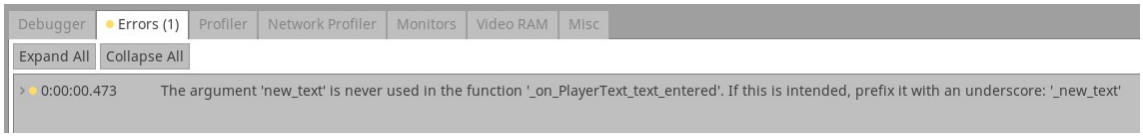
```
10
11 func _on_PlayerText_text_entered(new_text):
12     add_to_player_words()
13
14 func _on_TextureButton_pressed():
15     add_to_player_words()
16
17 func add_to_player_words():
18     player_words.append(PlayerText.text)
19     PlayerText.clear()
20     check_player_words_length()
21
22 func is_story_done():
23     return player_words.size() == prompts.size()
24
25 func check_player_words_length():
26     if is_story_done() :
27         tell_story()
28     else :
29         prompt_player()
30
31 func tell_story():
32     DisplayText.text = story % player_words
33
34 func prompt_player():
35     DisplayText.text = "Silahkan memasukkan " + prompts[player_words.size()]
```

Tugas 4.2

Tambahkan intro di awal game, kerjakan di laboratorium

E. Queue_free( ) dan get_tree( )

Dari game yang telah dibuat sejauh ini, jika setelah story ditampilkan, dan player/user memasukkan input maka akan terjadi error. Seperti pada Gambar 4.2



Gambar 4.2 Pesan error

Terdapat beberapa pilihan untuk mengakhiri game. Jika tell_story telah dikerjakan, apa yang selanjutnya akan dilakukan? Dapat mengelola input ke-5 atau menghapus Player_text atau mengakhirinya dengan cara lain. Agar game berakhir dengan baik silahkan tambahkan function untuk mengakhiri game.

Tugas 4.3

1. Tambahkan function end_game( )

2. Function end_game akan dipanggil ketika selesai mengerjakan story. Sehingga function ini akan dipanggil di line terakhir function tell_story( )
3. Jika end_game( ) maka hapus node tersebut dari dari frame saat ini

```
40 func end_game():
41     PlayerText.queue_free()
```

4. Cari dan Analisa mengenai queue_free( ) di dokumentasi Godot
5. Namun button akan tetap bekerja. Silahkan kelola untuk TextureButton

Mungkin anda telah melakukan editing pada script anda sebelumnya seperti potongan code berikut ini:

```
func _on_TextureButton_pressed():
        if is_story_done() :
                end_game()
        else :
                add_to_player_words()
```

Dalam hal ini kita dapat menggunakan method pada class Node, yaitu get_tree, dimana get_tree( ) akan mengembalikan nilai berupa objek SceneTree yang terdiri dari Node tersebut. Berbeda dengan get_node( ), get_node( ) hanya mengembalikan node saat itu saja, namun get_tree( ) akan mengembalikan keseluruhan node sejak dari Node LoonyLips

Tugas 4.4

6. Perbaiki terlebih dahulu code yang telah anda buat. Kali ini ketika end_game( ) maka ulangi lagi ceritanya
7. Tetap aktifkan button nya. Mungkin anda akan membuat function end_game menjadi sebagai berikut ini :

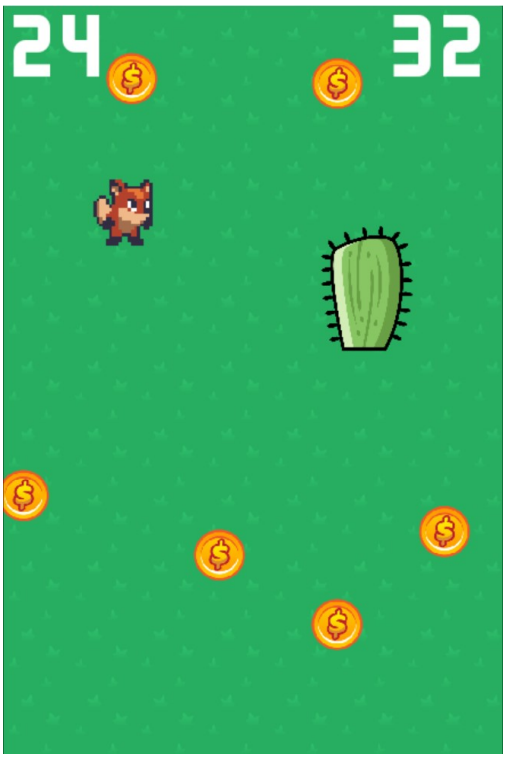
```
func end_game():
        PlayerText.queue_free()
        tell_story()
```

8. Coba jalankan game anda
9. Buatlah label yang berbeda, jika end_game, maka Label akan berisi “ulangi”
10. Di function ready tambahkan :
    PlayerText.grab_focus()
11. Analisa method grab_focus

F. Mekanik game 2D – Coin Dash

Selanjutnya anda akan membuat game 2 dimensi Coin Dash. Secara singkat, mechanics game adalah :

1) Terdapat button start untuk memulai
2) Game dengan visualisasi 2 dimensi, seperti pada Gambar 4.3



Gambar 4.3 Game Coin Dash

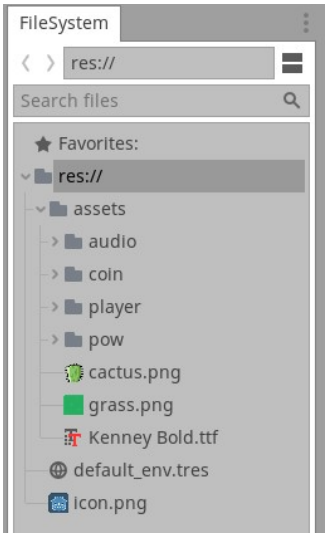
3) Tokoh dapat bergerak ke kanan kiri atas dan bawah
4) Tokoh bergerak untuk mengumpulkan coin di dalam area bounding box
5) Area bounding box memiliki background rumput hijau
6) Coin di letakkan di dalam area secara random setiap levelnya
7) Setiap player berkolisi dengan coin, akan menimbulkan efek suara, dan coin menghilang seiring dengan pertambahan skor
8) Setiap coin habis, akan naik level dan melakukan random peletakkan coin di level berikutnya
9) Challenge timer
10) Skor akan ditampilkan di kiri atas layar
11) Waktu ditampilkan di kanan atas layar
12) Akan diberikan sebuah objek (dalam hal ini kaktus), jika player berkolisi dengan kaktus maka game over
13) Akan tampil game over jika player mengenai kaktus
14) Akan tampil juga button start di halaman game over

Itulah mechanics dari game yang akan dirancang. Dari mechanics akan terlihat algoritma seperti apa yang harus dirancang, serta pengaturan layar dan assets-assets yang dibutuhkan

G. Membuat project baru Coin Dash

Tugas 4.5

1. Silahkan membuat project baru untuk game Coin Dash ini. Dalam project baru ini, anda akan membuat 3 scene independent yaitu Player, Coin dan HUD. Dimana akan dikombinasikan ke dalam satu game dalam scene Main
2. Letakkan seluruh resources sehingga akan terlihat seperti pada Gambar 4.4

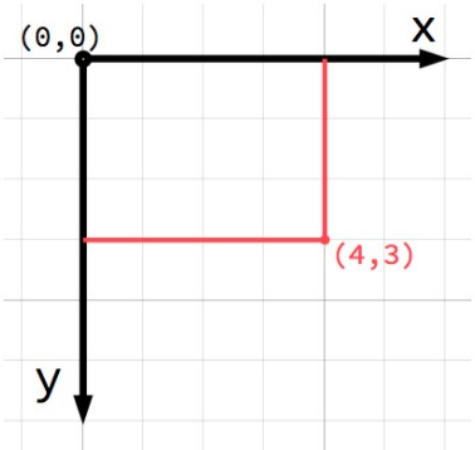


Gambar 4.4 Isi folder res

3. Atur project menjadi game portrait. Klik menu Project >> Project Settings. Pilih Display >> Window dan atur ukuran width 480 dan height 720. Kemudian atur Stretch/Mode → 2D, dan Aspect → Keep

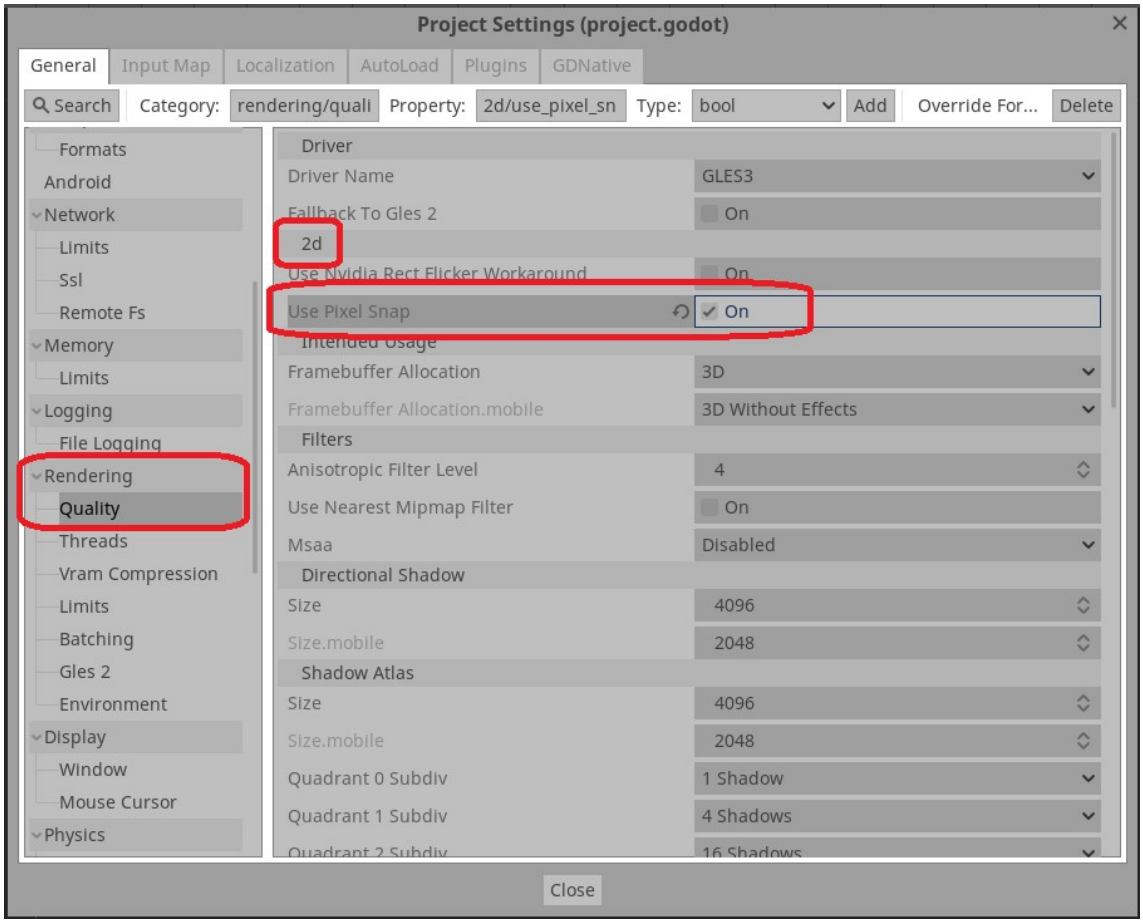
H. Vektor di Godot

Seperti yang telah dijelaskan di kelas perkuliahan, bahwa pengaturan koordinat adalah seperti pada Gambar 4.5



Gambar 4.5 Sistem Koordinat di Godot

Di dalam Godot terdapat vertor 2 dimensi dan 3 dimensi. Nama kelas masing-masing diwakili oleh **Vector2** dan **Vector3**. Vector di Godot menggunakan nilai floating point, bukan Integer. Artinya sebuah vector 2 dimensi bisa bernilai (1.5, 1.5), karena objek bisa digambarkan sebagai ½ pixel, ini akan menyebabkan masalah pada game pixel art. Untuk mengatasi hal ini, silahkan lakukan pengaturan pada pada Project Settings >> Rendering → Quality → 2D → Use Pixel Snap : On. Seperti pada Gambar 4.6



Gambar 4.6 Mengatur masalah pada pixel floating point

Jika anda membuat game 2 dimensi, sebaiknya lakukan selalu pengaturan ini.

I. Membuat Scene Player

Scene pertama yang akan dibuat yaitu yang akan mendefinisikan objek Player. Salah satu manfaat membuat Scene terpisah adalah Anda dapat mengujinya secara mandiri, bahkan sebelum Anda membuat bagian lain dari game. Pemisahan objek game ini akan menjadi semakin bermanfaat seiring bertambahnya ukuran dan kompleksitas project Anda. Memisahkan objek game individu satu sama lain membuatnya lebih mudah untuk memecahkan masalah, memodifikasi, dan bahkan mengganti seluruhnya tanpa memengaruhi bagian lain dari game. Termasuk juga

membuat karakter pemain agar sesuai, dan mendeteksi tabrakan dengan objek lain dalam game.

Tugas 4.6

1. Buatlah node bertipe Area2D dan rename menjadi Player
2. Lalu save scene Player
3. Sebelum menambahkan child node apa pun, sebaiknya pastikan Anda tidak memindahkan atau mengubah ukurannya secara tidak sengaja dengan mengkliknya. Pilih node Player dan klik ikon di sebelah kunci seperti pada Gambar 4.7

Gambar 4.7 Mengunci node

Area2D

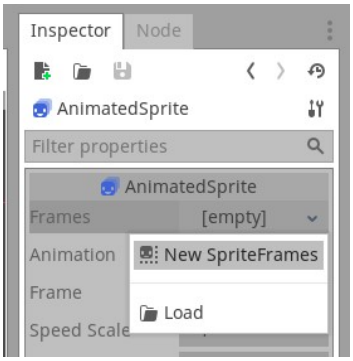
Dengan menggunakan node Area2D, Anda dapat mendeteksi ketika objek lain tumpang tindih atau menabrak pemain, tetapi Area2D sendiri tidak memiliki penampilan apapun, sehingga harus diberikan **AnimatedSprite** ke dalam dirinya

4. Tambahkan node AnimatedSprite sebagai child dari node Player

AnimatedSprite

AnimatedSprite akan menangani tampilan dan animasi untuk Player. Perhatikan bahwa ada simbol peringatan di sebelah node. Sebuah AnimatedSprite membutuhkan sumber daya **SpriteFrames**, yang berisi animasi yang dapat ditampilkannya.

5. Untuk membuatnya, temukan properti Frame di Inspector dan klik New SpriteFrames seperti pada Gambar 4.8

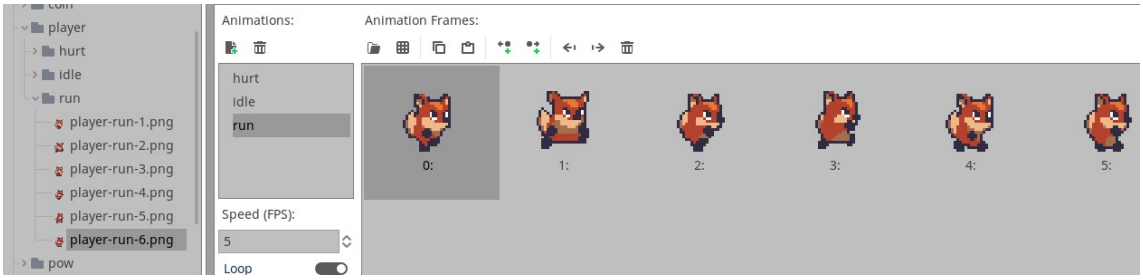


Gambar 4.8 Menambahkan New SpriteFrame

6. Kemudian pada tempat yang sama, klik SpriteFrame untuk membuka panelnya seperti pada Gambar 4.9

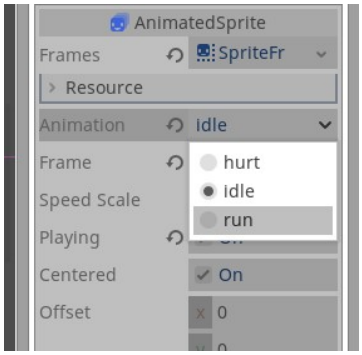
Gambar 4.9 Panel SpriteFrame

7. Di bagian sebelah kiri berisi daftar animasi. Klik **default** dan ganti namanya menjadi **run**.
8. Kemudian, klik tombol **Add** dan buat animasi kedua bernama **idle** dan animasi ketiga bernama **hurt**.
9. Di dok FileSystem di sebelah kiri, temukan gambar player run, idle, dan hurt, lalu drag and drop ke animasi yang sesuai seperti pada Gambar 4.10.



Gambar 4.10 Menambahkan gambar ke panel

10. Setiap animasi memiliki pengaturan kecepatan default 5 frame per detik. Ini agak terlalu lambat, jadi klik masing-masing animasi dan atur pengaturan Kecepatan (FPS) ke 8.
11. Di **Inspector**, pada property **Playing**, centang On. Dan pilih **Animation** untuk melihat animasi yang sedang dilakukan seperti pada Gambar 4.11



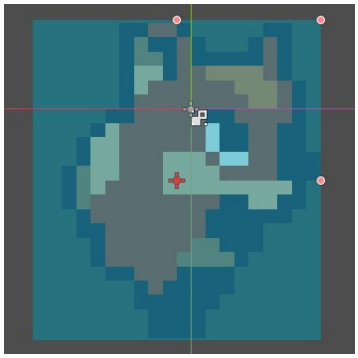
Gambar 4.11 Klik Animation untuk melihat animasi yang telah dibuat

Collision Shape

Saat menggunakan **Area2D**, atau salah satu collision object lain di Godot, perlu menentukan bentuk area kolisi, jika tidak ditentukan maka tidak dapat mendeteksi tabrakan. Sebut saja Namanya collision shape. Collision shape menentukan wilayah yang ditempati objek dan digunakan untuk mendeteksi tumpang tindih dan / atau tabrakan. Bentuk ini didefinisikan oleh **Shape2D**, dan mencakup persegi panjang, lingkaran, poligon, dan jenis bentuk lainnya.

Saat Anda perlu menambahkan sebuah bentuk ke area atau physic body, dapat ditambahkan **CollisionShape2D** sebagai child node. Anda dapat memilih jenis bentuk yang dibutuhkan dan diedit ukurannya di editor.

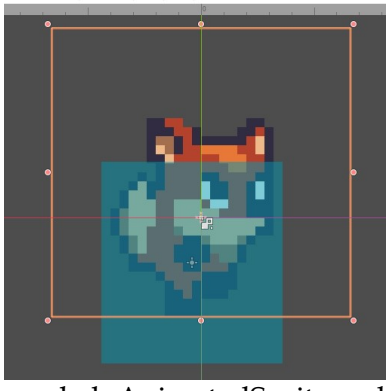
12. Tambahkan **CollisionShape2D** sebagai child node dari Player (bukan child dari **AnimatedSprite**). Ini akan memungkinkan Anda untuk menentukan hitbox pemain, atau batas area tabrakannya.
13. Di **Inspector**, di samping **Shape**, klik <null> dan pilih **New RectangleShape2D**. Sesuaikan ukuran bentuk untuk menutupi sprite seperti pada Gambar 4.12



Gambar 4.12 Shape yang akan digunakan untuk Collision Shape

Perhatikan bentuk tumbukan tidak berpusat pada sprite. Itu karena sprite itu sendiri tidak berada di tengah secara vertikal.

14. Perbaikilah dengan menambahkan offset kecil ke **AnimatedSprite**. Klik AnimatedSprite cari properti Offset di Inspector. Atur ke (0, -5). Setelah selesai, adegan Pemain Anda akan terlihat seperti Gambar 4.13. atur kembali Collision shape.



Gambar 4.13 Gambar rubah AnimatedSprite sudah berada di tengah

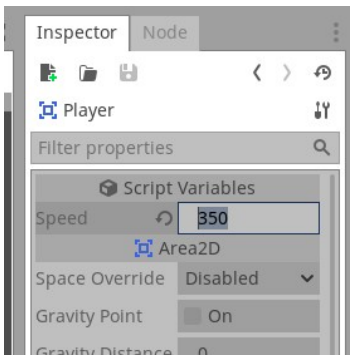
J. Menambahkan Script pada Player

Silahkan tambahkan script pada node Player

Program 4.4 Player.gd

```
1  extends Area2D
2  export (int) var speed
3  var velocity = Vector2()
4  var screensize = Vector2(480, 720)
```

Extends menjelaskan bahwa node ini diturunkan dari class Area2D. Line 2 terdapat script export, memungkinkan Anda menetapkan nilainya di Inspector, serta memberi tahu Inspector jenis data apa yang harus dimuat dalam variabel. Ini sangat berguna untuk nilai yang ingin disesuaikan, sama seperti Anda menyesuaikan properti bawaan node. Klik pada node Player dan atur properti Speed menjadi 350, seperti yang ditunjukkan pada Gambar 4.14. Sementara itu screensize akan digunakan untuk mengatur batas pergerakan pemain.



Gambar 4.14 Variabel Speed akan tampil di Inspector

K. Menggerakan Player

Berikutnya akan digunakan function **_process( )** yang berfungsi untuk menentukan apa yang akan dilakukan pemain. Function _process( ) dipanggil di setiap frame, jadi Anda akan menggunakannya untuk mengupdate elemen game yang Anda harapkan akan sering berubah. Anda membutuhkan pemain untuk melakukan tiga hal:

1) Periksa input keyboard
2) Bergerak ke arah yang diberikan
3) Mainkan animasi yang sesuai

Pertama, Anda perlu memeriksa input. Untuk permainan ini, Anda memiliki empat input arah untuk diperiksa (empat tombol panah). Aksi input didefinisikan pada **Project Setting** di bagian tab **Input Map**.

Anda dapat mendeteksi apakah sebuah input ditekan menggunakan **Input.is_action_pressed( )**, yang mengembalikan nilai true jika tombol ditekan dan false jika tidak. Menggabungkan status keempat tombol akan memberi Anda arah pergerakan yang dihasilkan. Misalnya, jika Anda menahan kanan dan bawah pada saat yang sama, vektor kecepatan yang dihasilkan adalah (1, 1). Dalam kasus ini,

karena kita menambahkan gerakan horizontal dan vertikal secara bersamaan, pemain akan bergerak lebih cepat daripada jika mereka hanya bergerak secara horizontal. Anda bisa mencegahnya dengan menormalkan kecepatan, yang berarti mengatur panjangnya menjadi 1, lalu mengalikannya dengan kecepatan yang diinginkan (seperti dibahas pada materi perkuliahan pertemuan ke 6). Lanjutkan program 4.4 seperti pada Program 4.5

Program 4.5

```
6  func get_input():
7      velocity = Vector2()
8      if Input.is_action_pressed("ui_left"):
9          velocity.x -= 1
10     if Input.is_action_pressed("ui_right"):
11         velocity.x += 1
12     if Input.is_action_pressed("ui_up"):
13         velocity.y -= 1
14     if Input.is_action_pressed("ui_down"):
15         velocity.y += 1
16     if velocity.length() > 0:
17         velocity = velocity.normalized() * speed
```

Program tersebut adalah pembahasan pada pertemuan mata kuliah berjudul "Chasing and evading". Lanjutkan Program 4.5

Dengan mengelompokkan semua kode ini bersama-sama dalam fungsi get_input( ), akan lebih mudah untuk mengubahnya nanti. Misalnya, Anda dapat memutuskan untuk beralih ke joystick analog atau jenis pengontrol lainnya. Panggil fungsi ini dari _process( ) lalu ubah **position** Player dengan **velocity** yang dihasilkan. Untuk mencegah Player keluar screem, dapat menggunakan function _clamp( ) untuk membatasi nilai **position** maksimum dan minimum. Lanjutkan Program 4.5

Program 4.5

```
19 func _process(delta):
20     get_input()
21     position += velocity * delta
22     position.x = clamp(position.x, 0, screensize.x)
23     position.y = clamp(position.y, 0, screensize.y)
```

Tugas Akhir

Save kemudian Run program, atau tekan F6

Jelaskan/analisa keseluruhan Program Player.gd setiap line-nyake dalam laporan tertulis