BAB IV

HASIL DAN PEMBAHASAN

4.1 Implementasi Sistem

Pada bab ini akan dijelaskan bagaimana sistem yang telah dirancang sebelum nya kemudian di implementasikan menjadi sebuah aplikasi utuh. Aplikasi yang di kembangkan ini digunakan untuk mengenalkan permainan tradisional Bentengan.

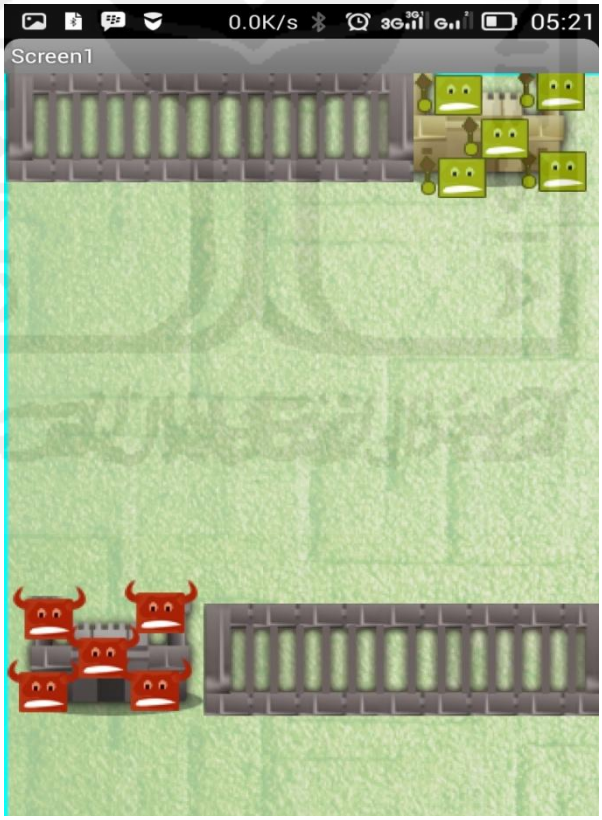
Dalam implementasi sistem ini terdiri dari beberapa tahap. Yaitu :

1. Implementasi pada aplikasi app inventor sebagai aplikasi utama dalam mengembangkan game tersebut.
2. Menguji program pada perangkat *smartphone* yang menggunakan sistem operasi android.
3. Menganalisis jalannya program.
4. Memperbaiki program jika terjadi kesalahan dalam berjalannya program.

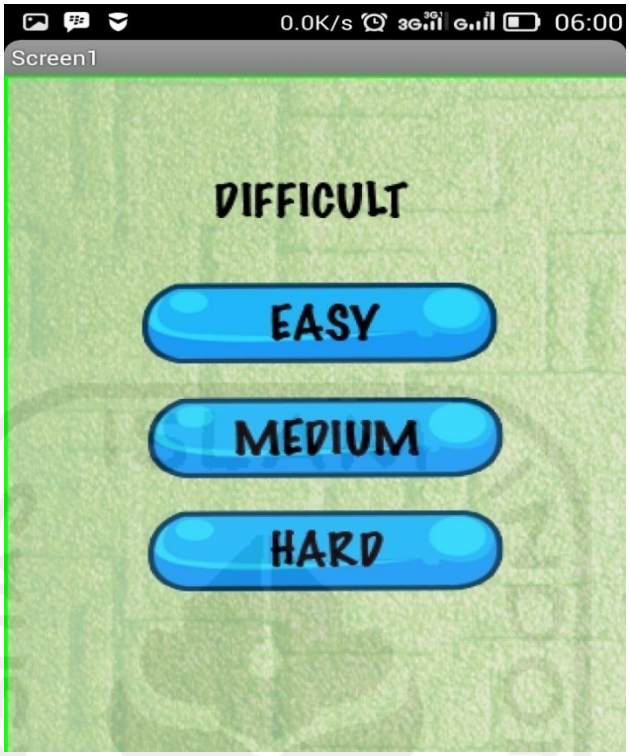
4.1.1 *Interface* Permainan

Pada *interface* permainan menunjukan hasil dari tampilan aplikasi game bentengan yang sudah jadi. *Interface* dari hasil akhir game tersebut berdasarkan dari rancangan tampilan permainan. Untuk *interface* aplikasi game bentengan dapat dilihat pada gambar 4.1, gambar 4.2, gambar 4.3 dan gambar 4.4

Gambar 4.1 Tampilan Home



Gambar 4.2 Tampilan Arena



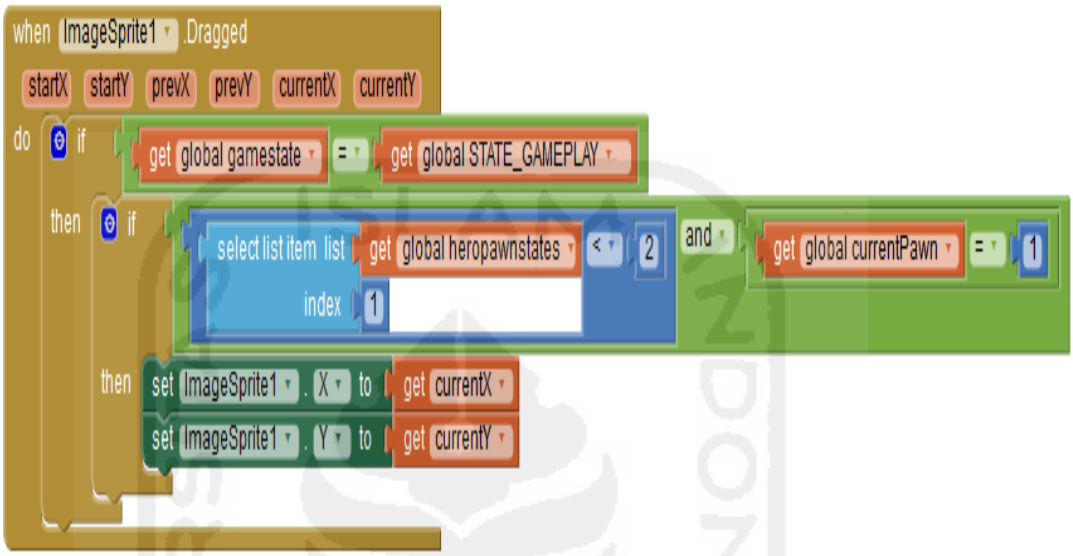
Gambar 4.3 Tampilan pilihan kesulitan

Tampilan pilihan kesulitan atau *Difficult* digunakan untuk mengatur kecepatan gerak dari *enemy*. Penentuan kecepatan gerak dari *enemy* ini saat berada di arena berdasarkan banyak pixel pada layar yang dilewati dalam satuan waktu. Untuk pilihan kesulitan *easy* kecepatan geraknya diseting 13 pixel. Untuk pilihan *medium* kecepatan geraknya diseting 20 pixel. Untuk pilihan kesulitan *hard* kecepatan geraknya diseting 30 pixel.

4.1.2 Implementasi Gerakan Player

Pada aplikasi ini untuk menggerakkan player menggunakan blok dari perintah *when.ImageSprite1.Dragged*. Pada blok ini karakter *ImageSprite1* dapat digerakkan sesuai dengan koordinat X dan Y pada arena. *ImageSprite1* di sini adalah karakter *hero* no 1 dari 5 karakter *hero* yang dimainkan oleh player dan setiap karakter memiliki pengaturan blok sendiri-sendiri. Pada setiap pengaturan gerak pada karakter *hero* 2,3,4 dan 5 tinggal dirubah kondisi dari

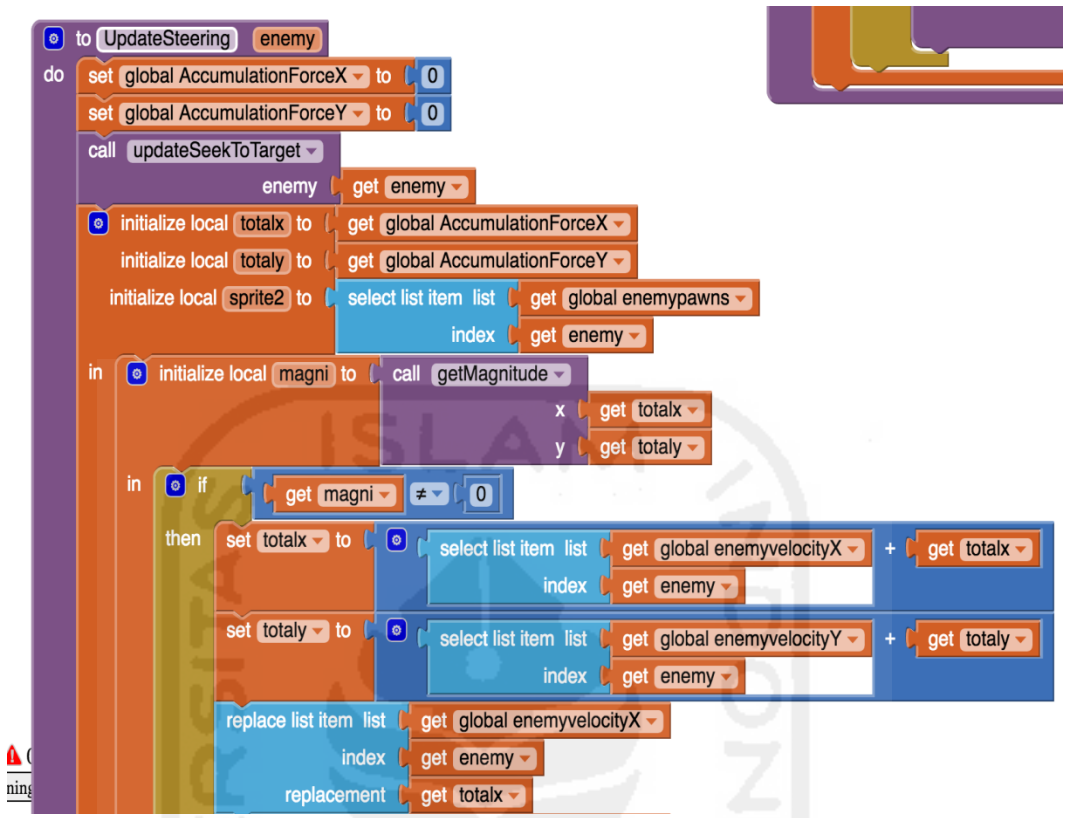
when.ImageSprite1.Dragged menjadi *when.ImageSprite2.Dragged* pada *hero* 2 , *when.ImageSprite3.Dragged* pada *hero* 3, *when.ImageSprite4.Dragged* pada *hero* 4, dan *when.ImageSprite5.Dragged* pada *hero* 5.

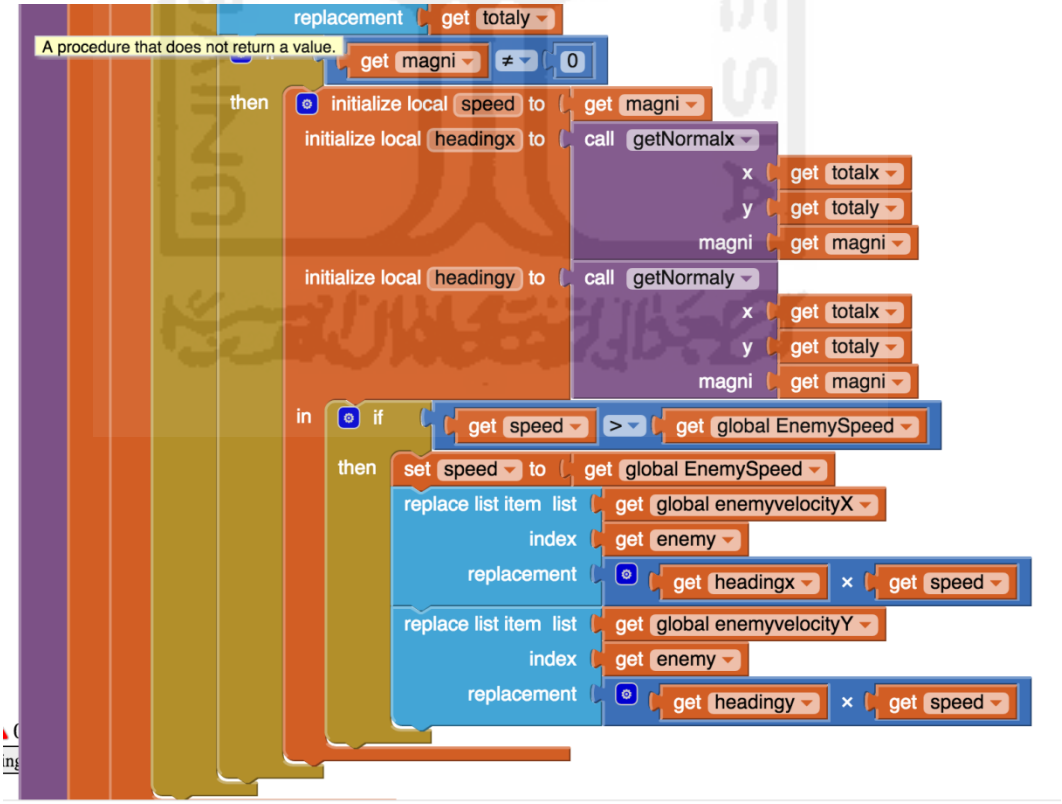


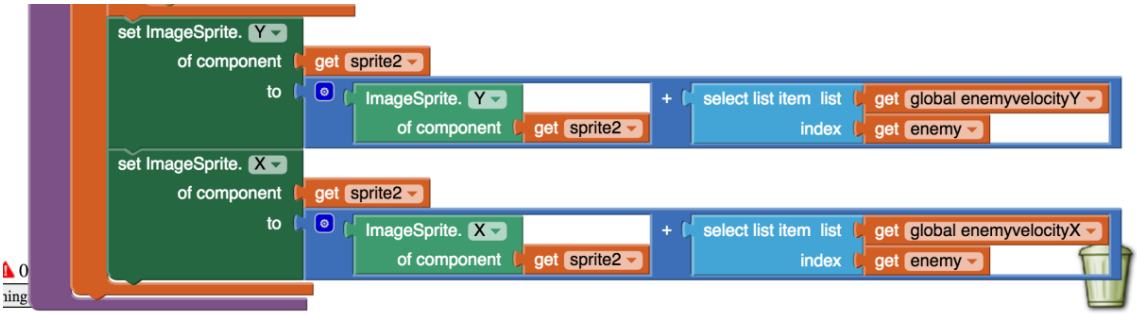
Gambar 4.4 Kode blok *when.ImageSprite1.Dragged*

4.1.3 Implementasi Gerakan Enemy

Pada aplikasi ini untuk menggerakkan *enemy* menggunakan *method* dari *to.UpdateSteering.enemy*. Pada method *to.UpdateSteering.enemy* berguna untuk menentukan arah dari gerakan *enemy* berdasarkan koordinat titik x dan y. Untuk menentukan update arah gerakan terdapat pada *method UpdateSeekToTarget.*


to UpdateSteering enemy
do
set global AccumulationForceX to 0
set global AccumulationForceY to 0
call updateSeekToTarget
enemy get enemy
initialize local totalx to get global AccumulationForceX
initialize local totaly to get global AccumulationForceY
initialize local sprite2 to select list item list get global enemypawns
index get enemy
in initialize local magni to call getMagnitude
x get totalx
y get totaly
in if get magni ≠ 0
then set totalx to select list item list get global enemyvelocityX + get totalx
index get enemy
set totaly to select list item list get global enemyvelocityY + get totaly
index get enemy
replace list item list get global enemyvelocityX
index get enemy
replacement get totalx


replacement get totaly
A procedure that does not return a value.
get magni ≠ 0
then initialize local speed to get magni
initialize local headingx to call getNormalx
x get totalx
y get totaly
magni get magni
initialize local headingy to call getNormaly
x get totalx
y get totaly
magni get magni
in if get speed > get global EnemySpeed
then set speed to get global EnemySpeed
replace list item list get global enemyvelocityX
index get enemy
replacement get headingx × get speed
replace list item list get global enemyvelocityY
index get enemy
replacement get headingy × get speed

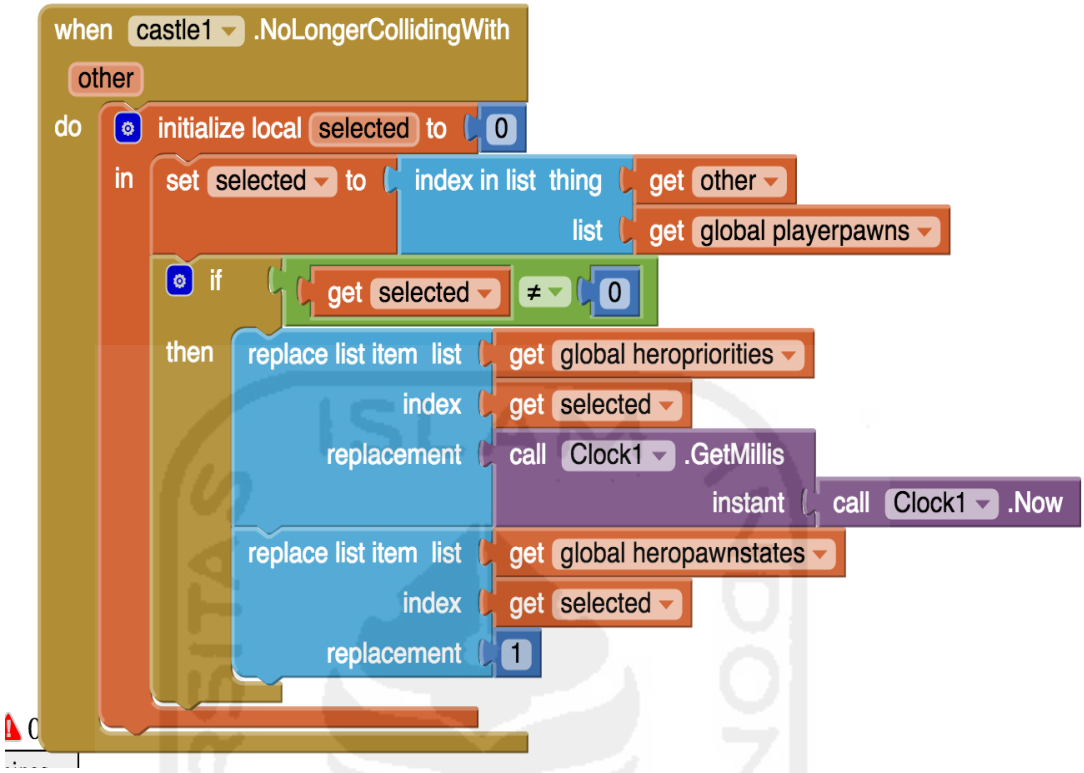


Gambar 4.5 Kode *block to.UpdateSteering.enemy*

4.1.4 Implementasi Prioritas Pemain

Pada arena bentengan ini di bagi menjadi 3 *state* yaitu *state* 0, 1 dan 2. Pada *state* 0 merupakan kondisi dimana para karakter *hero* atau juga *enemy* berada pada posisi di dalam benteng atau masih menyentuh benteng. *State* 1 adalah kondisi di mana posisi karakter *hero* atau *enemy* keluar dari benteng nya dan masuk ke arena. Pada posisi ini di sebut dengan posisi tempur. sedangkan *State* 2 adalah kondisi dimana *hero* atau *enemy* berada di dalam penjara. Untuk penentuan prioritas menurut *state* maka *state* 0 prioritas nya lebih tinggi daripada *state 1* dan *state* 1 prioritas nya lebih tinggi dari *state 2*. *Method* yang digunakan untuk penentuan *state* tersebut terdapat pada blok *when.castle1.NoLongerCollidingWith. method when.castle1.NoLongerCollidingWith* digunakan untuk pemain *hero* sedangkan untuk *enemy* menggunakan blok *when.castle2.NoLongerCollidingWith*

Pada saat *hero* atau *enemy* pada state 1 atau mode bertempur, pengaturan prioritasnya didasarkan akan waktu. Semakin kecil lama waktu pada kondisi *state* 1 maka prioritas nya semakin besar.

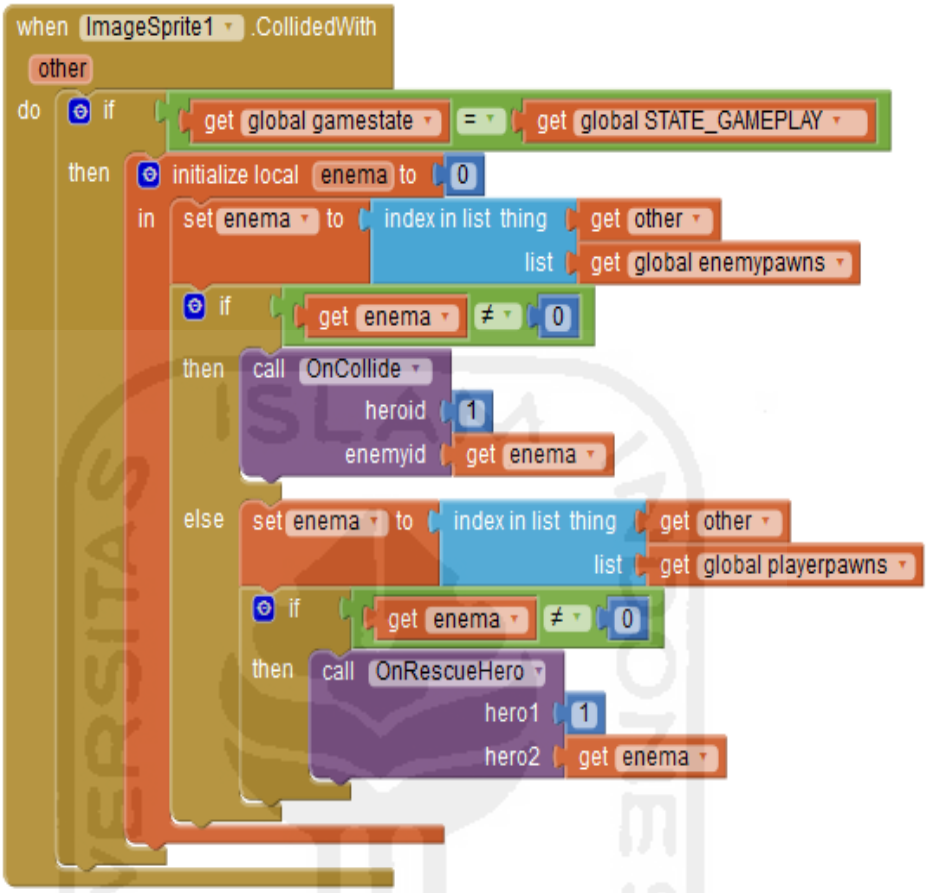


Gambar 4.6 Kode *block when.castle1.NoLongerCollidingWith*

4.1.5 Implementasi Tabarakan *Hero*

Pada permainan bentengan ada kondisi dimana saat *hero* bersentuhan dengan *hero* lain. Ketentuan deteksi tabrakan ini dibagi menjadi 2 kondisi yaitu tabrakan dengan kawan atau tabrakan dengan *enemy*. Jika kondisi nya tabrakan dengan kawan maka tidak terjadi apa-apa namun jika tabrakan dengan *enemy* maka akan di cek status prioritas nya siapa yang lebih tinggi seperti pada implementasi prioritas *hero* untuk menentukan siapa yang masuk ke penjara (*Prison*).

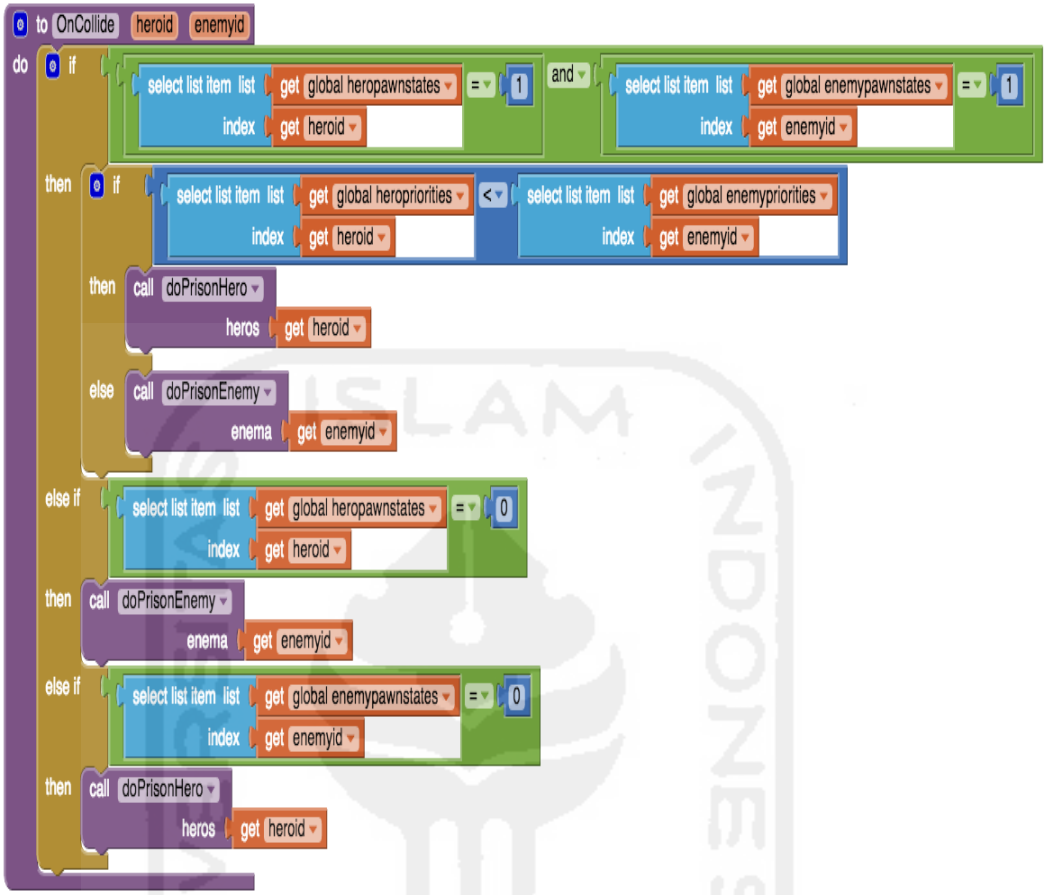
Pada aplikasi ini pola deteksi di atur pada blok *Collidedwith*. Sebagai contoh seperti pada gambar 4.7 untuk karakter *hero* 1 (*ImageSprite1*) menjadi *when.ImageSprite1.Collidedwith.* Karena di tim player ada 5 pemain, jadi untuk setiap karakter *hero* yang lain mempunyai pengaturan blok sendiri tinggal merubah karakter *ImageSprite1* menjadi *ImageSprit2* dan seterusnya.



Gambar 4.7 Kode blok *when.ImageSprite1.Collidedwith*

4.1.6 Implementasi *Oncollide*

Merupakan sebuah method yang digunakan untuk mengecek prioritas pemain mana yang lebih tinggi antara *hero* dan *enemy* saat mereka bertabrakan. Penentuan prioritas ini di gunakan untuk menentukan pihak mana yang di penjara *enemy* atau *hero*. *Method* yang digunakan dalam penentuan prioritas saat tabrakan terdapat pada gambar 4.8.



Gambar 4.8 Kode blok *to.OnCollide.heroid.enemyid*

4.1.7 Implementasi Memasukkan Penjara

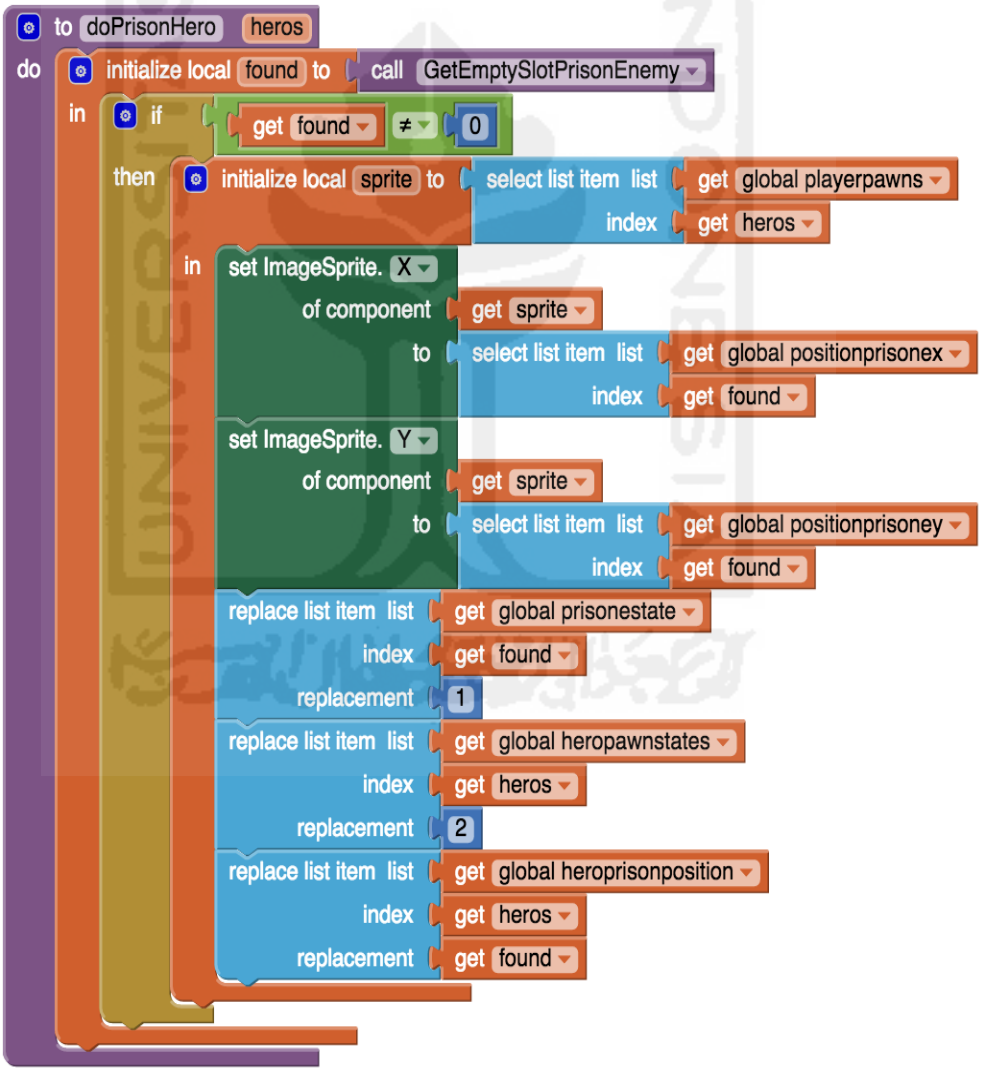
Pada permainan bentengan ada kondisi di mana *hero* atau *enemy* harus masuk ke penjara. Ada 2 kondisi yang menentukan untuk masuk ke dalam penjara yaitu, jika saat *hero* dan enemy bertabrakan dan prioritas *hero* lebih tinggi dari pada *enemy* maka *enemy* masuk ke dalam *PrisonEnemy*. Kemudian jika prioritas *enemy* lebih tinggi daripada *hero* maka *hero* masuk ke dalam *PrisonHero.*

4.1.7.1 *Prison Hero*

Pada aplikasi *Prison Hero* yaitu suatu kondisi dimana *hero* bertabrakan (*Collide*) dengan *enemy* yang memiliki prioritas lebih tinggi dari *hero* maka *hero* tersebut dimasuk kan ke dalam penjara. Pada blok di tulis berupa kondisi

global_heropriorities < *global_enemypriorities* maka *call doPrisonHero*. *doPrisonHero* merupakan sebuah blok perintah untuk menentukan posisi atau koordinat *Hero* di penjara dan merubah *state hero* dari angka 1 yang merupakan kondisi di mana *hero* keluar dari markas (posisi bertempur) menjadi angka 2 yang merupakan kondisi di saat dalam penjara.

Untuk penentuan posisi *hero* saat dipenjara menggunakan *method to.doPrisonHero.heros.* Pada *method* tersebut digunakan untuk menentukan koordinat *hero* di penjara berdasarkan koordinat x dan y.

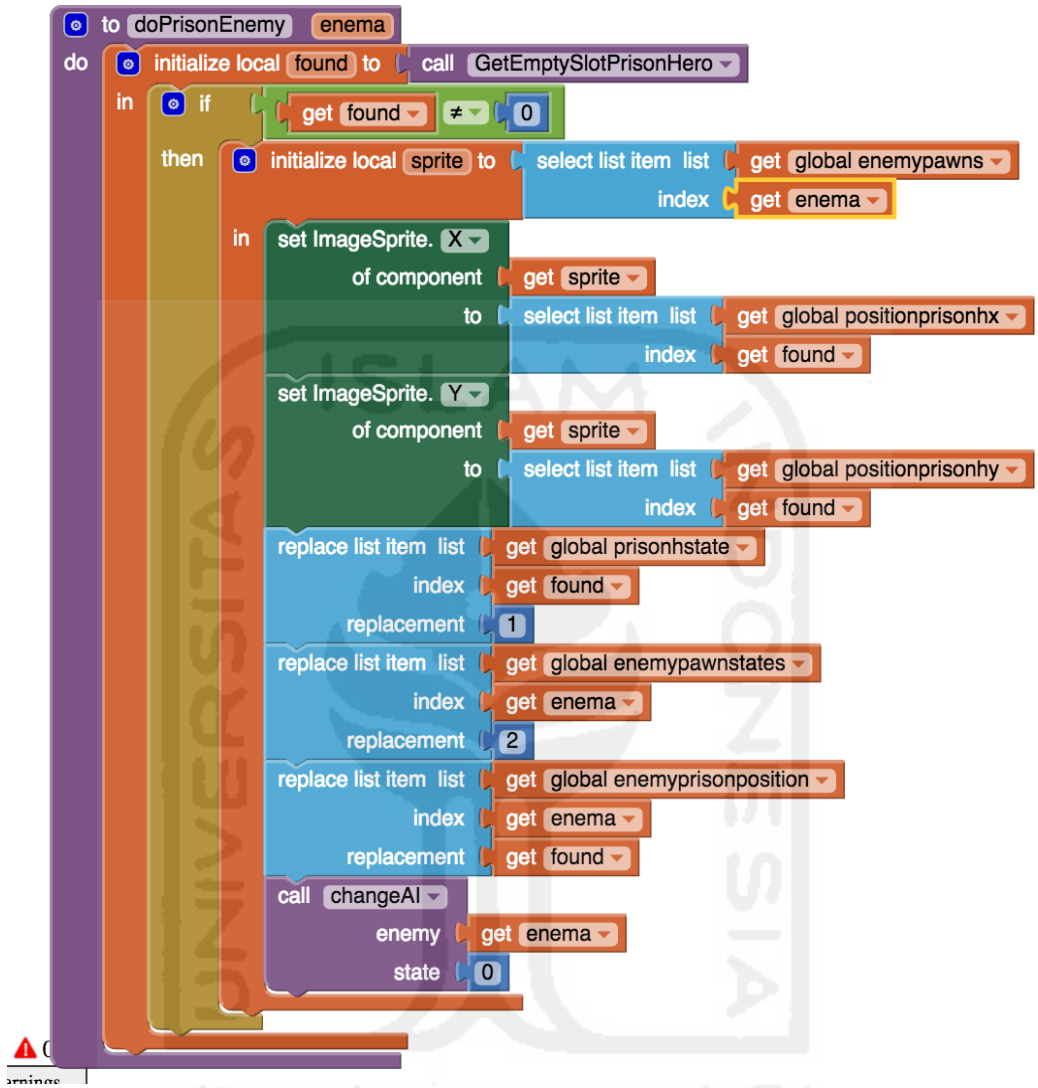


Gambar 4.9 Kode blok *to.doPrisonHero.heros*

4.1.7.2 *Prison Enemy*

Pada aplikasi *Prison Enemy* yaitu suatu kondisi dimanan *hero* bertabrakan (*Collide*) dengan *enemy* dan prioritas *hero* lebih tinggi daripada *enemy* maka *enemy* tersebut dimasuk kan ke dalam penjara. Pada blok di tulis berupa kondisi *global_heropriorities* > *global_enemypriorities* maka *call* doPrisonEnemy. *doPrisonEnemy* merupakan sebuah blok perintah untuk menentukan posisi atau koordinat *enemy* di penjara dan merubah *state enemy* dari angka 1 yang merupakan kondisi di mana *enemy* keluar dari markas (posisi bertempur) menjadi angka 2 yang merupakan kondisi di saat dalam penjara.

Untuk penentuan posisi *enemy* saat dipenjara menggunakan *method to.doPrisonEnemy.enema.* Pada *method* tersebut digunakan untuk menentukan koordinat *enemy* di penjara berdasarkan koordinat x dan y.



Gambar 4.10 Kode blok *to.doPrisonEnemy.enema*

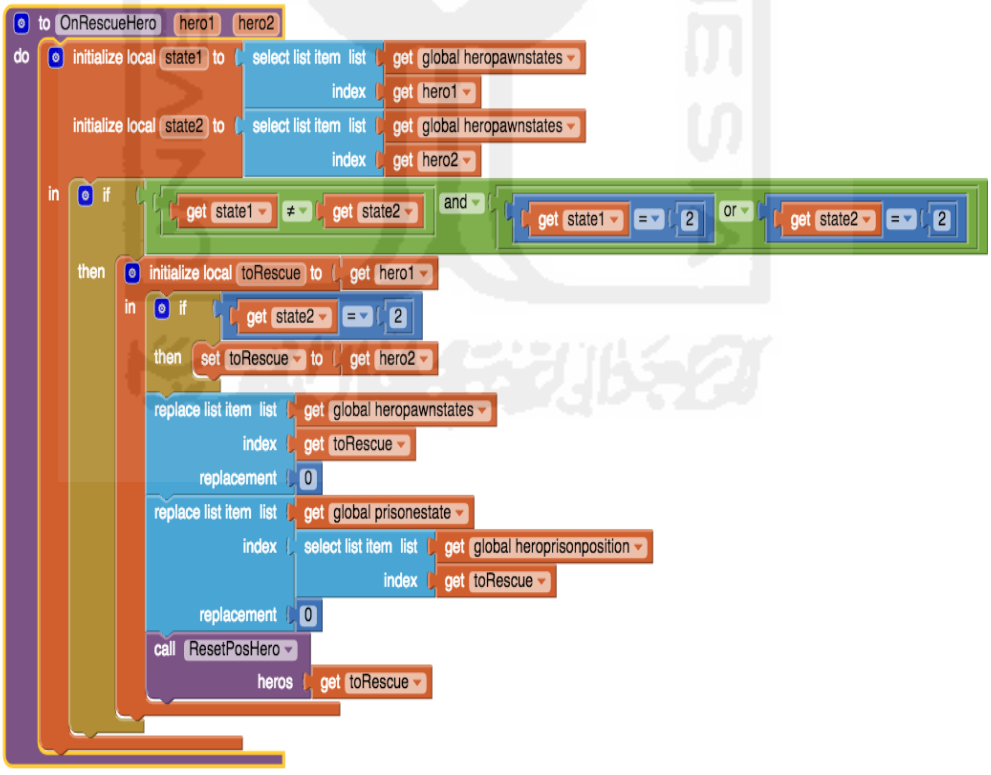
4.1.8 Implementasi Keluar Penjara

Keluar dari penjara merupakan sebuah kondisi dimana *hero* membebaskan *hero* lain yang tertangkap atau di penjara oleh *enemy* dengan bersentuhan dengan *hero* yang dipenjara tanpa tertangkap oleh *enemy*. Begitu juga sebaliknya dimana *enemy* membebaskan *enemy* lain yang telah tertangkap oleh pemain.

4.1.8.1 Pada Hero

Pada aplikasi ini *hero* berusaha membebaskan *hero* lain yang telah tertangkap oleh *enemy* dan di penjara di karena *hero* telah bertabrakan atu *collide* dengan *enemy* yang prioritasnya lebih tinggi dari pemain. Cara membebaskan *hero* yang dipenjara adalah dengan cara menyentuh *hero* tersebut dengan *hero* yang masih bebas dengan syarat *hero* tersebut dalam kondisi *state* 1 dan belum tersentuh oleh *enemy* dengan prioritas yang lebih tinggi.

Method yang digunakan untuk membebaskan *hero* dari penjara ada pada blok *to.onRescueHero.hero1.hero2*. Setelah di bebaskan, maka *hero* yang di penjara tersebut posisi nya di reset dan kembali ke posisi awal di markas. *State* awal *hero* waktu di tangkap yang memiliki *state* 2 kemudian di rubah menjadi 0, karena kembali ke markas. Untuk lebih detil tentang *method* *to.onRescueHero.hero1.hero2* dapat dilihat pada gambar 4.11.

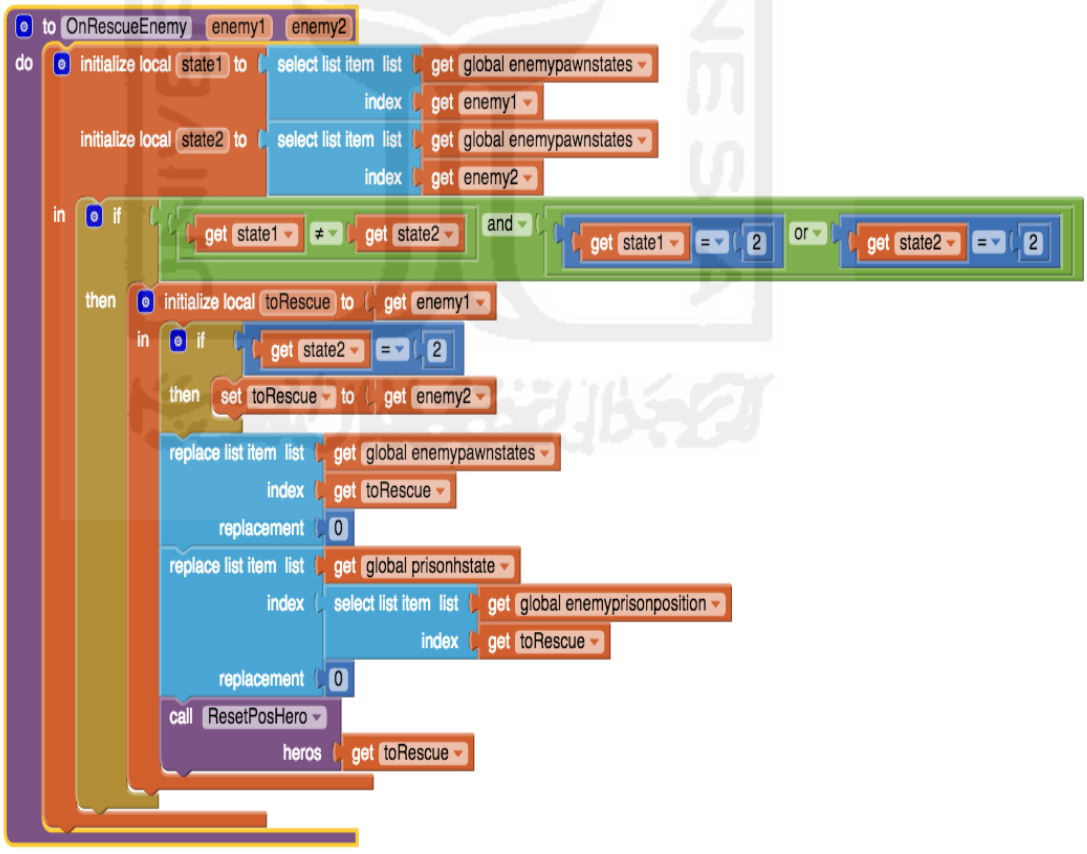


Gambar 4.11 Kode blok *to.OnRescueHero.hero1.hero2*

4.1.8.2 Pada Enemy

Pada aplikasi ini tim *enemy* membebaskan *enemy* yang tertangkap oleh *hero* milik kita dan di penjara di *PrisonEnemy*. Cara membebaskan *enemy* yang dipenjara adalah dengan cara menyentuh *enemy* tersebut dengan *enemy* yang masih bebas dengan syarat karakter *enemy* tersebut dalam kondisi *state* 1 dan belum tersentuh oleh *hero* dengan prioritas yang lebih tinggi.

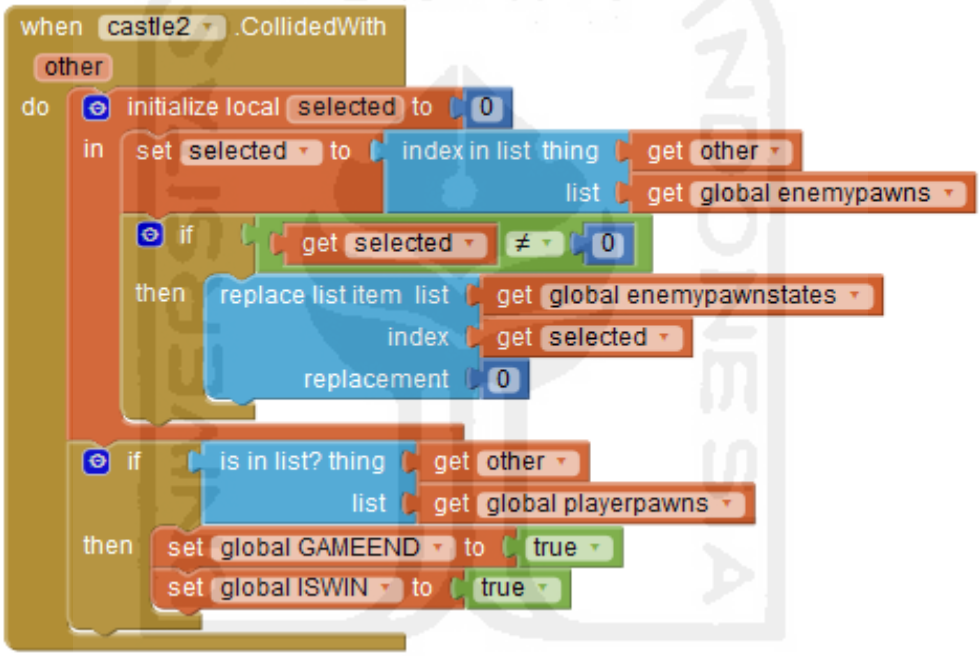
Method yang digunakan untuk membebaskan *enemy* dari penjara ada pada blok *to.onRescueEnemy.enemy1.enemy2*. Setelah di bebaskan, maka *enemy* yang di penjara tersebut posisi nya di reset dan kembali ke posisi awal di markas. *State* awal *enemy* waktu di tangkap yang memiliki *state* 2 kemudian di rubah menjadi 0, karena kembali ke markas. Untuk lebih detil tentang *method* *to.onRescueEnemy.enemy1.enemy2* dapat dilihat pada gambar 4.13.



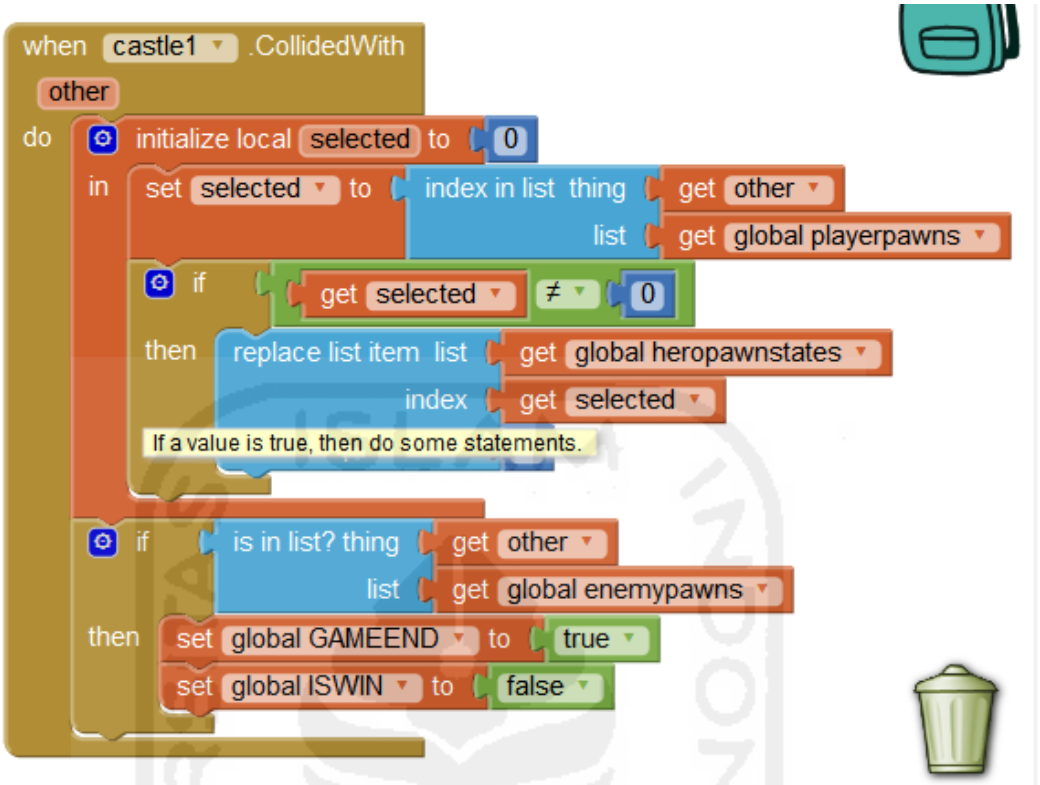
Gambar 4.12 Kode blok *to.OnRescueEnemy.enemy1.enemy2*

4.1.9 Kondisi Menang dan Kalah

Pada permainan bentengan terdapat kondisi menang dan juga kalah. *Method* yang digunakan ada 2 yaitu kondisi saat *Hero* menyentuh benteng *enemy* maka status nya menang. Sedangkan saat *enemy* menyentuh benteng *hero* maka status kalah. Untuk kondisi *hero* menyentuh benteng *enemy* menggunakan blok *when.castle2.Collidedwith* . Sedangkan untuk kondisi *enemy* menyentuh benteng *hero* menggunakan *method* blok *when.castle1.CollidedWith*.



Gambar 4.13 Status menang dengan blok *when.castle2.collidedwith*



Gambar 4.14 Status kalah dengan blok *when.castle1.collidedwith*

4.2 Pengujian Aplikasi

Pengujian aplikasi game Bentengan ini menggunakan *Smartphone* Android dengan merek Lenovo bertipe A859. Untuk tampilan menu-menu aplikasi dapat dilihat pada lampiran yang terdiri dari tampilan Home atau menu utama, Tampilan pilahan *Sound* posisi Off dsan On, tampilan pemilihan tingkat kesulitan, tampilan saat menang dan juga tampilan saat kalah.

Tampilan menu utama dapat dilihat pada gambar L.1, pada menu utama pemain dihadirkan beberapa pilihan menu. Pilihan menu tersebut antara lain tombol *Play* untuk memulai permainan , tombol Sound On/Off untuk menghidupkan atau mematikan suara *backsound*, tombol *Difficult* untuk masuk ke dalam pemilihan tingkat kesulitan dan juga terdapat tombol *Exit*.

Tampilan saat tombol *Play* di pencet dan masuk ke dalam arena permainan dapat dilihat pada gambar L.2. Pada arena terdapat 5 pemain *Hero* dan 5 pemain

Enemy. Disini player dapat menggerakkan pemain *hero* untuk mengalahkan *enemy*. Player menggerakkan pemain *hero* yang bertujuan untuk menyentuh benteng *enemy* tanpa tertabarak pemain *enemy*.

Tampilan menu pemilihan *Difficult* dapat dilihat pada gambar L.3. Pada tampilan menu terdapat 3 pilihan tingkat kesulitan yaitu *easy*, *medium* dan *hard*. Tingkat kesulitan yang dipilih berpengaruh terhadap tingkat kecepatan gerak *enemy*.

Saat player dan enemy berada di arena seperti pada gambar L4. Tampilan pada saat pemain enemy dan player berada di penjara seperti pada gambar L.5 Saat player menang dan berhasil mengalahkan *enemy* maka akan muncul status menang seperti pada gambar L.6 dan saat player kalah terhadap *enemy* maka akan muncul status kalah seperti pada gambar L.7.