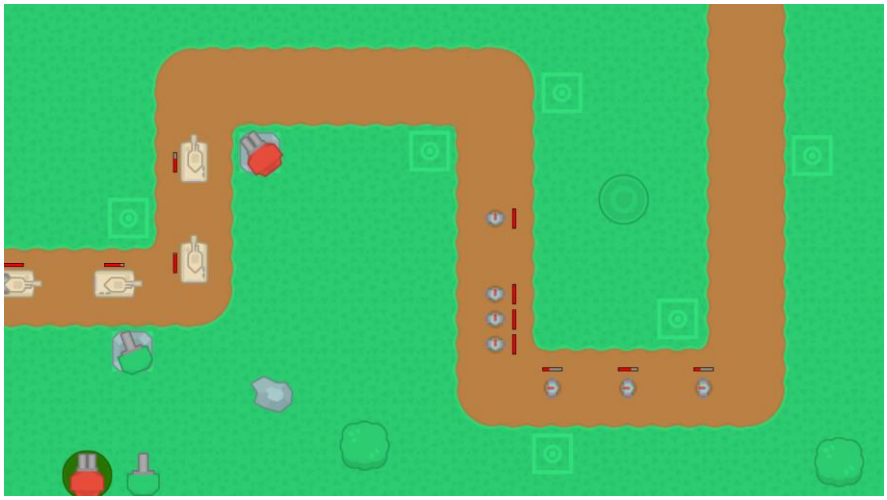
Unity Tower Defense

Af: Michael Hansen, Coding Pirates Furesø, 2022, version 0.50

Dokument og kode ligger her: https://github.com/mhfalken/unity/



Dette er en guide i hvordan man laver et simpelt Tower Defense spil i Unity, som vist på billedet. Det forudsætter lidt kendskab til Unity, som svarer til at man har lavet 2D Platform spillet, som er beskrevet i unity-intro.pdf. Det er også meningen at man skal bruge det omtalte dokument til opslag når der er noget man ikke kan huske eller finde ud af.
Spillet er bevist lavet simpelt så det er lettere at lave, dvs. der er steder hvor koden kunne være lavet mere optimal, men dermed også mere kompleks.

Dokumentet og koden er lavet i Unity version 2021.3. Den burde også virker i andre versioner, men der kan være små forskelle.

1 Guide

Denne guide er bevist lavet mere overordnet, da det er meningen at man skal lave mere selv og dermed komme op på et højere Unity niveau.

1.1 Import start pakke

Opret et 2D game og importer https://github.com/mhfalken/unity/blob/main/tower_defense.unitypackage.
Under *Graphics* ligger en masse grafik billeder som man kan bruge af. Hvis man mangler nogle, så må man selv finde dem på nettet.
Start med at vælge **Level1** under **Scenes**.

1.2 Kort og fjender

1. Start med at tegne en bane med en vej igennem under *Grid/Terrain* (brug **Tile Paletten**.)
2. Tag et billede af en fjende og sæt det ved start af vejen (uden for billedet). Kald det noget 'fornuftigt'.
3. Tilføj en **Collider 2d** og sæt **Trigger**. Sørg for at collideren er lidt større end figuren, da skuddene senere ellers har svært ved at ramme.
4. Tilføj **Rigidbody 2D** og sæt **Gravity Scale** til 0 og **Collision Detection** til **Continuous**.
5. Tilføj et **Tag** som hedder *Enemy*.
6. Tilføj scriptet *Enemy* til objektet.
7. Kør spillet og se hvad der sker.
8. Åben *Enemy* scriptet.

I `Update()` er der en `if` sætnings som gør at *fjenden* skifter retning for at følge vejen. Denne sætning skal rettes så *fjenden* følger den vej I har tegnet. For hvert sving skal der bruges en `if` sætning, så *fjenden* følger hele vejen. Her er en lille stump af koden fra *Enemy* script filen.

```
if ((transform.position.x > -20) && (path == 0))
{
    transform.rotation = Quaternion.Euler(0, 0, 90);
    path++;
}
else if ((transform.position.y > 11.5f) && (path == 1))
{
    transform.rotation = Quaternion.Euler(0, 0, 0);
    path++;
}
```

Man skal primært rette de ting som står med **rød fed**. Sidste tal i `Euler(0, 0, z)` er drejningen af figuren i grader.
Når man er færdig kan det være en fordel at lave en **PreFab** af fjenden.

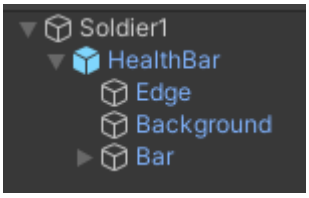
1.3 Health bar

For at vi kan skyde nogle forskellige fjender, har vi brug for nogle parametre til at styre hvor kraftige skud vi bruger og hvor meget et skud skader vores fjender før de dør.

Egenskab	Forklaring	Variabel navn
Hit impact	Hvor kraftig kuglen er. Dette tal starter ved 100.	hitImpact
Hit divider	Hvor meget fjenden kan klare. Dette tal starter ved 10.	hitDiv
Speed	Hvor hurtig fjenden bevæger sig.	Speed

Health	Hvor meget liv fjenden har tilbage. Det er et tal i intervallet [0-100]	Health

For hvert skud som rammer udregnes skaden med følgende formel:
health = health - hitImpact/hitDivider
Når health er nul eller negativ, så dør fjenden.
Vi skal nu lave en health bar så man kan se hvor meget liv vores fjende har tilbage. Selve health bar'en er lavet som en **Prefab**, så den skal 'bare' tilsluttes til vores fjende.



Åben fjende **PreFab**'en (dobbelt klik på den) og tilføj *HealthBar* **PreFab**'en til den så det ser sådan ud (min fjende hedder Soldier1):

Tryk på *Soldier1* og træk *Bar* objektet over i feltet **Bar Obj**. Flyt health bar'en så den står et godt sted fx ovenover figuren. Gå ud af **PreFab** mode.
Selve koden til at styre health bar'en er allerede lavet i *Enemy* scriptet og ligger i funktionen `Hit()`.
Kør spillet og se at alt virker.
Placer en lille række af fjender ud for start af vejen, så har man lidt at teste med.

1.4 Kanonkugle

Vi skal nu lav en kugle, som vores kanon kan skyde med.

1. Tag billedet af kuglen (*Bullet*) og træk det ind i **Scenen**.
2. Tilføj en **Collider 2D,** lav den lidt større end kuglen og sæt **Trigger**.
3. Tilføj scriptet *BulletCtrl.*

Sæt kuglen et 'godt sted' og se at den virker, når den rammer en fjende (health bar skal tælle ned og kuglen skal forsvinde).
Lav en **PreFab** når det virker.

1.5 Tårne

Vi skal nu lave et forsvarstårn og få det til at virke.

1. Opret et tårn med en kanon og kald det *Tower*.
2. Tilføj en **Collider 2D** til kanonen som skal være så stor som kanonen kan skyde og sæt også **Trigger**.
3. Tilføj *Tower* scriptet til kanonen.
4. Tilføj *Bullet* fra **PreFabs** til **Bullet Obj** (Kanons **Inspector**)

Se om det virker, ved at få en fjende til at bevæge sig inden for rækkevidde af tårnet.
Tårnet kan skyde, men kanonen drejer ikke. Tilføj følgende linjer til *Tower* scriptet:

```
float angle;

angle = AngleCalc(Vector3.up, dir);
transform.rotation = Quaternion.Euler(0, 0, angle);
```

Se at kanonen drejer inden den skyder.
Juster skudkraft og skudhastighed i **Inspector**'en og lav en **PreFab** af tårnet.
Her vil det være fint at lave nogle flere forskellige fjender og lave **PreFabs** af dem.

1.6 Platforme

Lige nu skal tårnene placeres inden spillet starter, hvilket jo ikke er meningen. Vi skal derfor lave nogle platforme som tårnene kan stå på.

1. Find et billede til platformen og træk det ind i **Scenen**. Kald det Platform.
2. Tilføj en **Collider 2D**
3. Tilføj *PlatformCtrl* scriptet
4. I **Inspector**'en tilføj et element i **Towers Obj** og træk tårn **PreFab**'en over i det.

Start spillet og se at man kan placere et tårn på platformen.
Man kan desværre placere flere tårne på samme platform. Ret koden så der kun kan stå et tårn på hver platform. **Hint**: Lav en variabel som husker om der står et tårn allerede.
Lav en **PreFab** og indsæt nogle flere platforme i spillet.

1.7 Game Master og tårn valg

Udfordringen med spillet er at der er behov for at holde styr på nogle resurser globalt set. Til det formål skal der oprettes en gamecontroller.

1. Opret et tomt objekt (**Create Empty**) og kald det *GameMaster*
2. Tilføj *GameMaster* scriptet

GameMaster scriptet skal bruges til at gemme globale variable, som kan tilgås fra alle de andre scripts. Lige nu indeholder det kun `public int towerSelected`.

Vi skal nu lave så man kan vælge mellem flere forskellige tårne. Først skal man lave nogle flere forskellige tårne, så der er noget at vælge imellem og tilføje dem til listen i *Platform* objektet (**Towers Obj**).

1. Find et billede af tårnet og træk det ind i **Scenen**
2. Tilføj en **Collider 2D**
3. Tilføj *TowerSel* scriptet
4. Ret **Tower Index** så det passer med listen i *Platform*. Første element i listen har indeks 0.

Gør det hele en gang til, men med et anden tårn billede og ret **Tower Index** så det passer.
Placer begge billeder ude i kanten af skærmen, så man kan bruge dem til at vælge hvilket tårn man vil indsætte.
Når man trykket på et af billederne, så bliver den globale variabel `towerSelected` sat til det valgte tårn.
Vi skal nu bruge den information til at indsætte det rigtige tårn på platformen, i stedet for bare at tage tårn 0.
Tilføj følgende linjer til *PlatformCtrl* scriptet.

```
GameMaster gameMasterObj;

gameMasterObj = GameObject.Find("GameMaster").GetComponent<GameMaster>();
```

Ret nu koden så den bruger variablen `gameMasterObj.towerSelected` i stedet for `0` til at vælge hvilket tårn man sætter ind.
Test at koden virker, ved at sætte forskellige tårne ind på forskellige platforme.

Når man har valgt et tårn at sætte ind, kan man ikke se hvilket tårn man har valgt. Tilføj nu noget kode, som viser hvilket tårn som man har valgt.
Hint: Man kan lave et billede (firkant/cirkel) som tårnet står på, og så få dette billede til at være synligt når tårnet er valgt. Til det formål kan følgende linjer være nyttige:

```
GameObject circleObj;
circleObj = gameObject.transform.Find("Circle").gameObject;
circleObj.SetActive(true);
```

Her hedder mit bilede objekt *Circle*.

1.8 Fjender dispenser

Vi har nu brug for at der kommer fjender hele tiden. Til det formål skal vi lave et objekt som genererer fjender.

1. Find et godt billedet til en dispenser (det vil kun kunne ses i **Scene** view ikke i selve spillet.)
2. Placer det ved start af vejen, uden for skærmen.
3. Tilføj scriptet *EnemyDispenserCtrl*
4. Tilføj alle fjenderne fra **PreFabs** til listen i **Inspector**'en (**Enemies Obj**).

Start spillet og se at der kommer fjender.
Det er generelt ikke nemt at lave en generator som laver fjender på en fed måde. Ret i koden, så der kommer flere forskellige typer fjender frem. Man kan eventuelt bruge følgende funktion, som laver et tilfældigt tal:

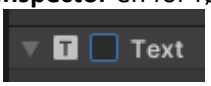
```
tal = Random.Range(MIN, MAX);
```

Hvor `tal` ligger fra og med MIN, til MAX, men uden MAX er med.
Matematisk skrives det således: `[MIN, MAX[`.
Eksempel: `Random.Range(4, 8)` vil lave et af følgende tal {4, 5, 6, 7}.
Det vil kræve nogle forsøg at få fjenderne til at komme på en rimelig måde.

1.9 Game Over

Hvis bare en fjende slipper igennem, så har man tabt.
Ændre koden i *Enemy* scriptet, så det skriver en tekst på skærmen, hvis en fjende slipper igennem. Det er lidt anderledes end 'normalt', da fjenderne bliver skabt on-the-fly og derfor ikke nemt kan linkes til teksten.

1. Opret en tekst som normalt og kald det *GameOver* og få det til at se godt ud.
2. For *GameOver* objekter, fjern fluebenet i **Inspector**'en for følgende felt (så feltet forsvinder):



3. I scriptet *Enemy*, indsæt følgende linje der hvor man har tabt spillet:

```
GameObject.Find("GameOver").GetComponent<Text>().enabled = true;
```

Det der sker er, at når man har tabt spillet, så bliver Game Over teksten synlig.
Prøv nu spillet og se om alt virker.

2 Næste skridt

2.1 Penge

Man får penge for at ødelægge fjender
Man bruger penge på at købe tårne og opgraderinger

2.2 Opgrader tårnene

Skud hastighed
Skud styrke
Rækkevidde
Opgrader health (se nedenfor)
Det koster penge at opgraderer

2.3 Fjender skyder

Fjender skyder
Kanonen drejer inden den skyder
Tårnet skal have en health bar
Platformen skal blive ledig når tårnet 'dør'