



| | | |
|---|---------------------------------------|---|
|  Institut Sabadell | UF4. Programació Orientada a Objectes |  Curs: 2019/2020 |
| | <i>Crea el teu Videojoc. Part 2</i> | |

Presentació

Un cop podem dibuixar Sprites, farem que es puguin moure, entre d'altres mètodes.

Punt A Input

Crea un nou paquet (Part2) i crea a dins una classe RocaControlable, filla de Roca, amb constructor estàndard (és a dir, el mateix que has fet servir a l'activitat anterior).

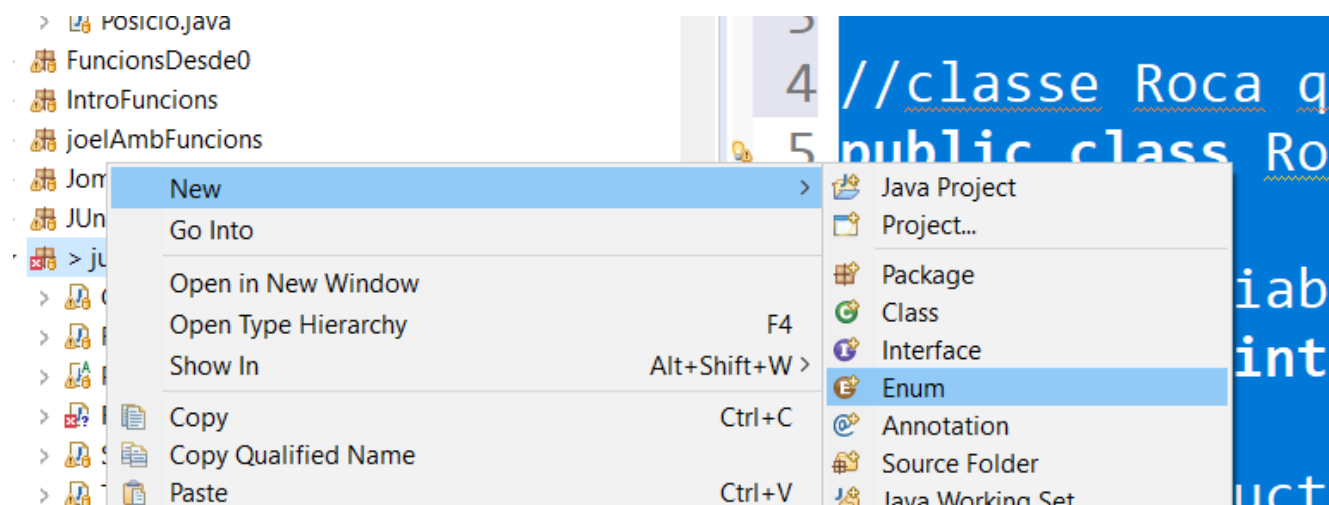
Fes també un main similar al de la Part1. Afegeix una funció *input()* al main amb el que controlaràs l'entrada de teclat.



Tens 3 funcions que controlen l'entrada de teclat, les tres desde Window. `window.getPressedKeys()`, que et torna les tecles que mantens apretades, `window.getKeysDown`, que et torna les tecles que apretes, només en el moment en que les apretes, i `window.getKeysUp` que te les torna quan les aixeques. Les tres tornen Sets (un tipus de llista que no et permet repetir continguts) amb el que si vols veure per exemple si tens la tecla 'd' apretada, es faria servir així

```
if(w.getPressedKeys().contains('d')) {
```

Crea un nou enum "Input" amb les opcions: DRETA, ESQUERRA, AMUNT, AVALL, GRAN, PETIT.

Què és un enum? Un enum



| | | |
|--|---------------------------------------|---|
|  | UF4. Programació Orientada a Objectes |  Curs: 2019/2020 |
| | <i>Crea el teu Videojoc. Part 2</i> | |

És un fitxer especial (no és una classe) que et permet fer una llista d'opcions de la que tries una. Aquí tens l'enum. Per que fer servir un enum en comptes de fer servir les opcions com a una String? Perquè així no fas typos i no t'equivoques escrivint.

```
public enum Input{
    DRETA,
    ESQUERRA,
    AMUNT,
    AVALL,
    GRAN,
    PETIT
}
```

fes a RocaControlable un mètode *moviment(Input in)* al que li passes un input i es mourà depenent de l'input entrat.

Un cop fet això hauràs de fer dues coses.



- En primer lloc, crear al main una funció *input(RocaControlable)* que mitjançant les funcions de teclat, cridi al mètode *moviment* de *RocaControlable* amb l'input adequat. Per exemple:

```
if(w.getPressedKeys().contains('d')) {
    rocacontrol.moviment(Input.AMUNT);
}
```

- Pensa si vols fer servir ifs o else ifs. Vols que la roca es mogui amunt i avall al mateix temps? i avall i a la dreta?
- En segon lloc, a *RocaControlable*, omplir el mètode *moviment*, amb el que cada opció faci això. Tots els mètodes es poden aplicar només tocant x1, y1, x2 i y2.

```
if(in == Input.AMUNT){
    //moure cap amunt tocant x1 i x2
}
```

- Què passa si mous x1 i x2 al mateix temps la mateixa distància? i si només mous x2?

| | | |
|---|---------------------------------------|---|
|  Institut Sabadell | UF4. Programació Orientada a Objectes |  Curs: 2019/2020 |
| | <i>Crea el teu Videojoc. Part 2</i> | |

- El mètode moviment hauria de decrementar en 1 el nombre d'accionsDisponibles. Si aquesta arriba a 0, pots esborrar la roca amb el mètode delete();

Executa al codi i comprova que pots controlar la roca.

Punt B Classes Abstractes

Ha arribat el moment de fer servir les físiques. Crea un nou main i una nova classe que es dirà Personatge. Aquesta classe Personatge extindrà de PhysicBody. PhysicBody es una classe filla de Sprite, amb el que heredes tots els mètodes de Sprite, però també té mètodes nous. Més important, PhysicBody es una classe abstracta, amb dos mètodes abstractes. Això significa que no pots instanciar un PhysicBody, com si fos un Sprite, però pots instanciar fills, sempre que implementin els mètodes abstractes

Implementa els mètodes abstractes en Personatge, però els pots deixar buits, de moment. Implementa també un enum, de nom Input2, amb tres opcions, DRETA, ESQUERRA, SALT.

Tenim 4 tipus de forces físiques. Velocitat, Força, Força Constant, i Fre (Drag). Parlarem de les tres primeres.

La velocitat es dona en x,y, i està mesurada en píxels/frame. És a dir, si poses una velocitat (-1,0), a cada frame es mourà en un píxel a l'esquerra.

La força està mesurada en píxels/frame². És a dir, una força de (0,-1), farà que en el primer frame es mogui un píxel a dalt, el següent 2 píxels a dalt, el següent 3 píxels, etc.



La força constant s'utilitza per forces que son constants, principalment la gravetat. Declara una força constant de (0,0.2) per a representar la gravetat. Això serà constant amb el que ho pots fer al constructor.

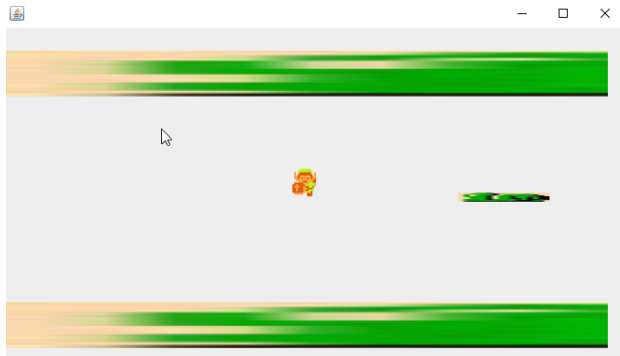
```
this.setConstantForce(0, 0.2);
```

Fixa't que tens per cada variable dos mètodes. set, que el modifica, i add, que afegeix al valor actual, com si fos un acumulador. Fes servir el què sigui adient en cada cas.

Sembla lògic que per moure's a esquerra i a dreta es faci servir una velocitat i per a saltar una força. Implementa la funció de moviment. Posa també un parell de plataformes. Pots aprofitar la classe Roca.

Executa el codi. T'hauria de quedar quelcom similar a això.

| | | |
|---|---------------------------------------|---|
|  Institut Sabadell | UF4. Programació Orientada a Objectes |  Curs: 2019/2020 |
| | <i>Crea el teu Videojoc. Part 2</i> | |



Pensa que si vols modificar la velocitat per anar a esquerra i dreta només vols modificar la velocitat x, no la y. Com faries el modificar la velocitat x deixant la y igual?

Punt C Col·lisions, instanceof

Us haureu fixat que podeu saltar totes les vegades que vulgueu, ja que no comproveu que només pogueu saltar si esteu tocant a terra. Anem a arreglar això. Creeu una variable booleana que sigui “aterra”. Només podreu saltar si aterra es true, i a més, cada cop que salteu aterra es posarà a false.

Fins aquí la part fàcil. La part difícil és: Com saber que has tornat a tocar a terra? Per això estan els dos mètodes abstractes. OnCollisionEnter saltarà automàticament cada cop que el vostre PhysicBody col·lisiona amb un altre Sprite, i OnCollisionExit saltarà al deixar de col·lisionar. Com sabem si hem col·lisionat amb un terra? Assumint que totes les plataformes son de Classe Roca, es pot fer així

```
@Override
public void onCollisionEnter(Sprite sprite) {
    //si el sprite es de classe Roca
    //aterra=true;
}
```

Com pots saber si un sprite es d'una classe? Amb la comanda **instanceof**.

Pensa que això no ho arregla al 100%. Per exemple, si xoca amb una plataforma per la part d'abaix, podria tornar a saltar. Hi ha moltes formes per arreglar això, com fer que cada plataforma tingui 2 sprites, un per dalt i un per baix, i només el de dalt compte, o fent que per a activar l'aterrar el punt més baix teu ha de ser més alt o igual que el punt més alt de la plataforma que utilitzem.

Fes servir l'après per a implementar un “Doble Salt”. És a dir, que un cop a l'aire puguis tornar a saltar, una vegada.