LevelSVG Manual

Installing Inkscape

Warning: The SVG parser **ONLY** supports SVG files generated by Inkscape. You should NOT use any other SVG editor to generate the SVG files.

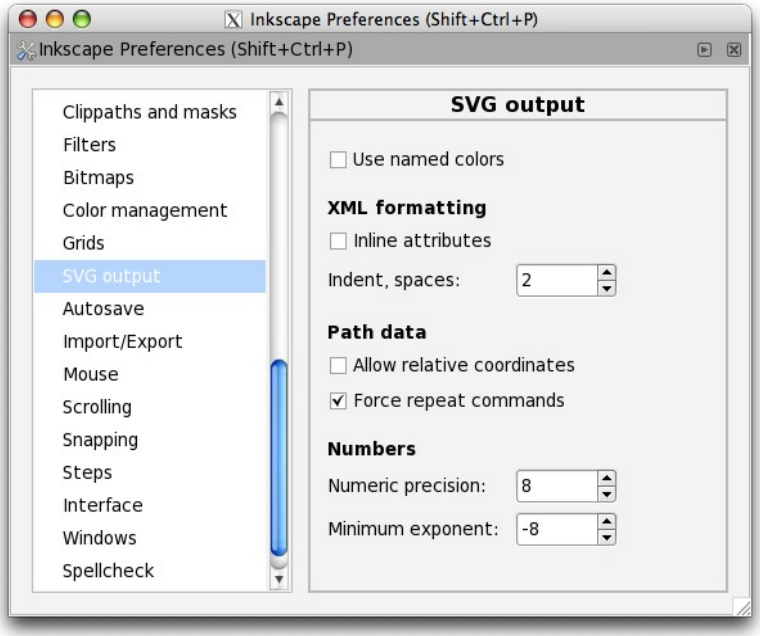
Download:

- http://www.inkscape.org/download/

Use version 0.47, 0.48 or newer. Older version are not supported.

Preferences:

1. Open Inkscape
2. Open Inkscape preferences (File → Inkscape Preferences)
3. Set the *SVG output* preferences
4. Path data
    5. Allow relative coordinates MUST be DISABLED
    6. Force repeat commands SHOULD be ENABLED for debugging purposes



Using Inkscape

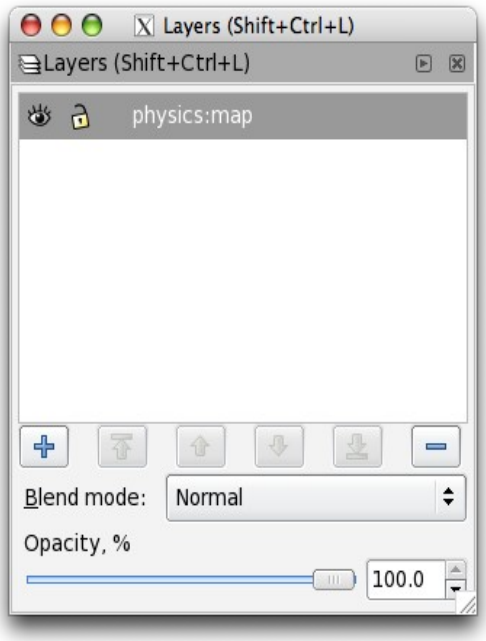
Layers

In your SVG file you can have as many layers as you want, but only the layers that begins with the

`physics:` name will be parsed.

Example of layers:

- `physics:static` (notice that “static” is a helper name)
- `physics:dynamic` (“dynamic” is also a helper name)
- `physics:enemies` (“enemies”, another helper name)
- `map` (this layer won't be parsed)



Supported SVG objects

Supported Inkscape objects:

- *Rectangles:* It will create a solid box2d rectangle (1 fixture). They support: *restitution*, *density*, *friction* and *isSensor*. Take into account that *rounded* boxes are not supported.
- *3D Boxes:* They are supported, but they are not very useful for 2d games.
- *Ellipses:* It will create a solid box2d circle (1 fixture). If the X-radius is different from the Y-radius, it will create a circle of radius: (rx+ry)/2. They support: *restitution*, *density, friction* and *isSensor*.

- *Stars:* It will create a body with multiple edges (multiples fixtures). Since these objects are not “solid” objects, *density* is not supported.
- *Spirals:* It will create a body with multiple edges (multiple fixtures). Since these objects are not “solid” objects, *density* is not supported.
- *Pencil:* Limited support. Warning: Only useful to create straight lines. If it is used as a 'freehand' pencil, it will create objects that will make box2d crash.
- *Pen:* It will create a body with multiple edges (multiple fixtures). Since these objects are not “solid” objects, *density* is not supported.
- *Calligraphy*: Limited support. Warning: Only works with very simple objects, otherwise it will create objects that will make box2d crash.
- *Text*: Not supported

Additional features:

- Each of the supported objects can be transformed (translated, rotated and/or scaled)
- Each of the supported objects can be grouped with other supported objects.
- Each of the supported objects can be joined (union) with other supported objects, thus, creating just one box2d object.

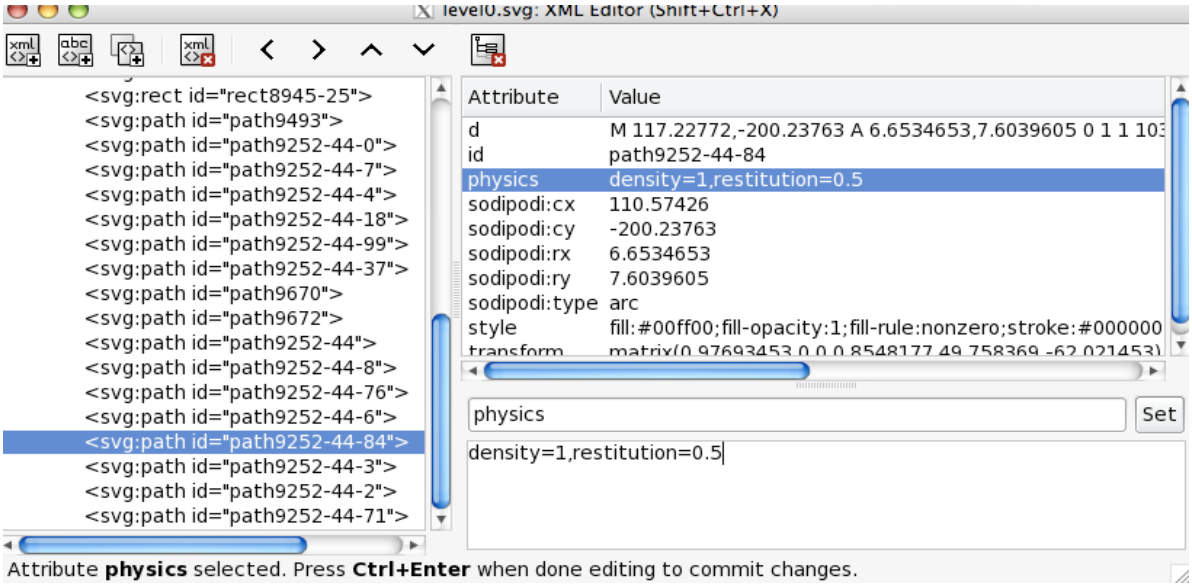
Integration with physics

For each SVG object a `b2Body` will be created. You can't create compound shapes in Inkscape, but you can do it from the “game_data” callback. See: *Integration with your game.*

Supported physics properties in the created SVG objects:

- `restitution`: float number. The restitution (elasticity) usually in the range [0,1].
- `density`: float number. If density is not present or if density is 0, the body will be treated as an static body. Otherwise it will be treated as a dynamic body. Usually in kg/m^2.
- `friction`: float number. The friction coefficient, usually in the range [0,1].
- `isSensor`: expects a boolean (0 or 1). Only supported on “solid” objects (circles and rectangles)
- `fixedRotation`: expects a boolean (0 or 1). If enabled, then the body won't rotate.
- `type`: an string. It could be “static”, “dynamic” or “kinematic”. By default it will be “static” if density is 0. If density is > 0 then, by default it will be “dynamic”.

To add any of these properties, you need to open the XML editor (Shift + Control + X).



You have to add the `physics` attribute to the elements that you want to edit.

The value is list of key-values separated with comas. Example of possible values:

```
density=0.3,friction=0.5,restitution=0.9
```

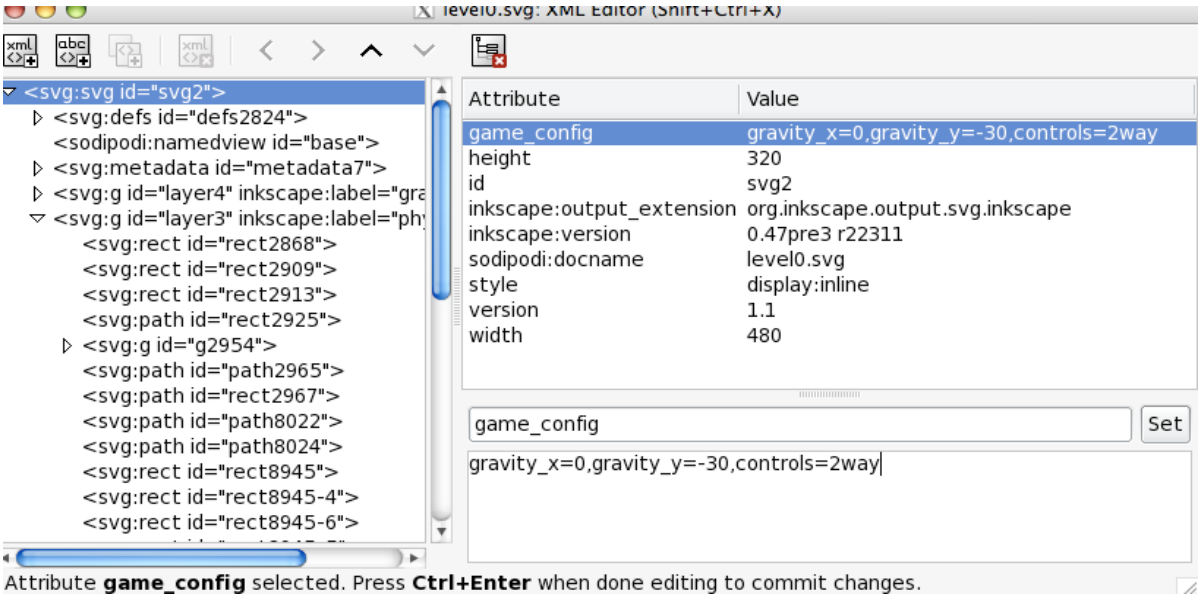
The order is not important.

Level configuration

You can also configure some level variables like:

- *gravity_x:* a *float* for the horizontal gravity (default is *0*)
- *gravity_y:*a *float* for the vertical gravity (default is *-10*)
- *controls*:
    - *2way:* Supports only horizontal movement of the hero
    - *4way:* Supports both horizontal and vertical movement of the hero (default)

To do so, you need to edit the 1st SVG element, in particular you need to edit the *<svg:svg>* element, and you need to add the *game_config* attribute to it:



Integration with your game

You can also link elements (box2d bodies) with cocos2d objects using the `game_data` attribute.

At present, the only supported keys are `object` and `objectParams` but you can extend it easily to match your needs.

Example:

```
game_data="object=portal,objectParams=moveToX:80;moveToY:1730"
```

Supported objects:

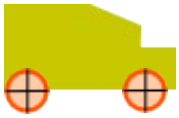
herobox:

- `object=herobox`
- It is a “Hero”, it means that it will create a main character. It can only be 1 hero per level.
- It is implemented in the `HeroBox.mm` file, with the class `Herobox`
- It is a subclass of `Hero.`
- It is implemented with 1 b2body with 1 boxed shape, with fixed rotation.
- It is a `dynamic` object.
- It is animated when it walks
- Supported parameters: n/a
- Used in levels: Level 2, Level 4 and Playground0

heroround:

- `object=heroround`
- It is a “Hero”, it means that it will create a main character. It can only be 1 hero per level.
- It is implemented in the `HeroRound.mm` file, with the class `Heroround`.
- It is a subclass of `Hero`.
- It is implemented with 1 b2body with a circular shape.
- It is a `dynamic` object.
- Supported parameters: n/a
- Used in levels: Level0, Level1, Level3 and Playground1

herocar:



- `object=herocar`
- It is a “Hero”, it means that it will create a main character. It can only be 1 hero per level.
- It is implemented in the `HeroCar.mm` file, with the class `Herocar`
- It is a subclass of `Hero`.
- It is implemented with several b2bodies and shapes. It is a complex box2d object.
- It is a `dynamic` object.
- Supported parameters: n/a
- Used in levels: Level5

princess:

- `object=princess`
- The “goal”. The hero needs to touch it in order to complete the level. There can be as many “princess” as you want.
- It is implemented in the file `Princess.mm` with the class `Princess`.
- It is a subclass of `BodyNode`
- It is implemented using 1 b2body and 1 circular fixture.
- It is a `dynamic` object.
- Supported parameters: n/a
- Used in levels: Level0, Level1, Level2, Level3, Playground0 and Playground1

princessbox:

- `object=princessbox`
- This is another "goal" object, like `Princess`.
- It is implemented in the file `PrincessBox.mm` with the class `PrincessBox`.
- It is a subclass of `Princess`
- It is implemented using 1 b2body and 1 box2d fixture.
- It is a `dynamic` object.
- The sprite is animated with several sprite frames.
- Supported parameters: n/a
- Used in levels: Level4 and Level5

enemy:

- `object=enemy`
- If the 'hero' touches any of these objects, he lose 1 life.
- It is implemented in `Enemy.mm` file in the class `Enemy`.
- It is implemented using 1 b2body with 1 circular fixture.
- It is a `dynamic` object.
- It is a subclass of `BadGuy`
- Supported parameters (`objectParams=patrolTime:0;patrolSpeed:2`)
  - `patrolTime` (float): the time it takes to go from left to right. Default: 0 (no movement)
  - `patrolSpeed` (float): the speed of the patrol (the speed is in Box2d units). Default: 2
- This object can by dragged by default.
- Used in levels: Level0, Level3, Level4

enemycar:

- `object=enemycar`
- Another kind of enemy, this time with the shape of a sort of car
- Supported parameters (`object=enemycar,objectParams=motorRPM:-250`)
  - motorRPM (float): The RPM of the "right" wheel. If the value is negative, it will move to the left. If it is positive, it will move to the right.
- Used in levels: level6

hole:

- `object=hole`
- Another kind of enemy. If the Hero touches any of these objects, it will lose 1 life.
- It is implemented in `Hole.mm` file in the class `Hole`.
- It is an `static` object.
- It is a subclass of `BadGuy`
- Used in levels: Level1

poison:

- `object=poison`:
- Similar to `hole`, but this time it will be an invisible object.
- Useful if you want to draw it using a TMX map.
- It is implemented in `Poison.mm` file in the class `Poison`.
- It is an `static` object.
- It is a subclass of `BadGuy`
- Used in levels: Level0 and Level4

platform1:

- `object=platform1`
- A One sided platform, AKA Super Mario platforms.
- It is implemented in the file `Platform1.mm` with the class `Platform1`
- It is a subclass of `BodyNode`
- It is an `static` object
- Supported parameters:
  - `visible (string):` default: “yes”
  - If you want to draw the platform using TMX maps, set `objectParams=visible:no`
- Used in levels: Level0, level2 and Level4

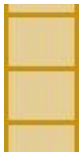
platform:

- `object=platform`
- A solid platform.
- It is implemented in the file `Platform.mm` with the class `Platform`
- It is a subclass of `BodyNode`
- It is an `static` object
- Used in levels: Level2.

movingplatform:

- `object=movingplatform`
- A platform that goes from horizontally or vertically.
- It is implemented in the file `MovingPlatform.mm` with the class `MovingPlatform`
- It is a subclass of `BodyNode`
- It is a `kinematic` object
- Supported parameters : `objectParams=direction:horizontal;duration:2;translation:150`
  - `direction (string)`: Possible values "`horizontal`" or "`vertical`"
  - `duration` (`float`): in seconds.
  - `translation` (`float`): in pixels
- Used in levels: level2 and level5

ladder:



- `object=ladder`
- A 'ladder' object. When the HeroBox touches it, the hero can climb up or down.
- It is implemented with 1 dynamic.
- This object is a `sensor`.
- It is implemented in the `Ladder.mm` file with the class `Ladder`
- It is a subclass of `BodyNode`
- Used in levels: Level4 and Level6

spinner:

- `object=spinner`
- An 'spinner' object. It is recommended that you create a SVG circle element to define the *spinner,* since the *spinner* will be as big as the circle.

- It is implemented with 1 dynamic body with 2 shapes, 1 static body and 1 revolute joint.
- This object can be rotated by touching it.
- It is implemented in the `Spinner.mm` file with the class `Spinner`
- It is a subclass of `BodyNode`
- Used in levels: Level3

portal:

- `object=portal`
- When the Hero touches this object, the hero will be teleported to a certain location
- It is implemented in the `Portal.mm` file with the class `Portal`
- It is a subclass of `BodyNode`
- It is an `static` object.
- Supported parameters: `(objectParams=moveToX:842;moveToY:800)`
    - `moveToX (float)`: X location in pixels
    - `moveToY (float)`: Y location in pixels
- Used in levels: level4 and level5

chain:

- `object=chain`
- A complex object implemented using 9 b2body objects, 8 fixtures and 8 joints
- 8 b2bodies are `dynamic` objects, while the “attach” b2body is an `static` one.
- It is implemented in the `Chain.mm` file with the class `Chain`
- It is a subclass of `BodyNode`
- Used in levels: level5 and playground0

fruit:

- `object=fruit`
- A bonus node. When touched by the Hero, it will increase the score by 1.
- It is a `sensor` and `static` object.
- It is implemented in the `Fruit.mm` file with the class `Fruit`
- It is a subclass of `BonusNode`
- Used in levels: level0, level1, level2, level3, level4, level5 and playground0.
- Supported parameters: (`objectParams=respawnTime:3`)
    - `respawnTime (float)`: If 0, then the it won't be respawned. Otherwise the

node will be respwawn N seconds after it has been touched by the hero

star:

- `object=star`
- A bonus node. When touched by the Hero, it will increase the score by 5.
- It is implemented in the `Star.mm` file with the class `Star`
- It is a subclass of `BonusNode`
- It is a `sensor` and `static` object.
- Used in levels: level0, level1, level2, level3, level4, level5 and playground0.
- Supported parameters: (`objectParams=respawnTime:3`)
    - `respawnTime (float)`: If 0, then the it won't be respawned. Otherwise the node will be respwawn N seconds after it has been touched by the hero

life:

- `object=life`
- A bonus node. When touched by the Hero it will increase the lives by 1.
- It is implemented in the `Life.mm` file with the class `Life`
- It is a subclass of `BonusNode`
- It is a `sensor` and `static` object.
- Used in levels: level0, level1, level2, level3, level4, level5 and playground0
- Supported parameters: (`objectParams=respawnTime:3`)
    - `respawnTime (float)`: If 0, then the it won't be respawned. Otherwise the node will be respwawn N seconds after it has been touched by the hero

Creating your own object

Subclassing BodyNode

You can easily add you own custom objects by editing extending the `BodyNode` class. Just create a subclass of `BodyNode` with the name that you want, and an instance of your class will be created each time the parser finds an object with it's name.

Example, when *object=enemy* is detected, an `Enemy` class will be instantiated.

WARNING: The class names MUST only have the initial character in capital letter.

Example of valid class names:

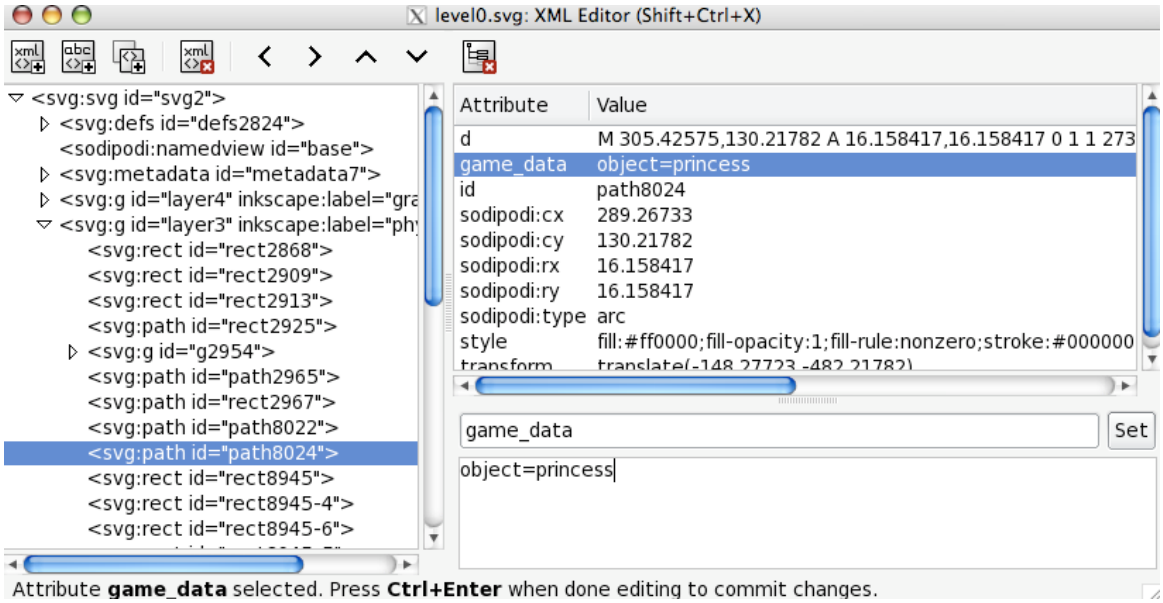
- `Herobox`
- `Heroround`
- `Enemy`

Example of invalid class names:

- `HeroBox`
- `HeroRound`
- `enemy`

How to create complex physics objects ?:

- Create complex box2d objects in Mekanimo or any other complex box2d editor. Coding complex box2d objects *by hand* (writing all the code) is also possible.
- In Inkscape create a simple object with more or less the dimensions of the complex object
- Add the *game_data* attribute to that object
- Create a subclass of `BodyNode`



For example, take a look at the 'spinner' object. It is a complex object that was coded by 'hand' but it's size and position are defined within Inkscape. See `Spinner.mm` file for further information.

BodyNode

The `BodyNode` class is the class that links Box2d objects with cocos2d objects.

So, in order to create a LevelSVG object you must subclass BodyNode.

BodyNode has the following properties:

- `reportContacts (int):` `(default: BN_CONTACT_NONE)`
  - `BN_CONTACT_NONE`
  - `BN_CONTACT_BEGIN`
  - `BN_CONTACT_END`
  - `BN_CONTACT_PRESOLVE`
  - `BN_CONTACT_POSTSOLVE`
  - or any combination like: `BN_CONTACT_BEGIN | BN_CONTACT_END`

- `isTouchable (BOOL):` `(default: NO)`
  - Is `YES`, then the object can be dragged by touching it
- `properties (int):` `(default: BN_PROPERTY_SPRITE_UPDATED_BY_PHYSICS)`
  - `BN_PROPERTY_NONE`
  - `BN_PROPERTY_SPRITE_UPDATED_BY_PHYSICS`
  - If you want to update the object using cocos2d (manually), then set the value `BN_PROPERTY_NONE`.

For example, lets see the `Enemy` class:

```
@implementation Enemy
-(id) initWithBody:(b2Body*)body game:(GameNode*)game
{
      if( (self=[super initWithSpriteFrameName:@"sprite_enemy_01.png"] ) ) {

         // bodyNode properties
         reportContacts_ = BN_CONTACT_NONE; // <---- NO CONTACTS WILL BE REPORTED
         body_ = body;
         isTouchable_ = YES;  // <----- THIS OBJECT CAN BE DRAGGED

         …

      }
      return self;
}

@end
```