GameL

The Game Scripting Language

Tianyu Cheng
Ben Lin
Mark Mansi

December 1, 2014

What is GameL?

- A scripting language

What is GameL?

- A scripting language
- Designed to abstract game constructs

What is GameL?

- A scripting language
- Designed to abstract game constructs
- Flexible

What is GameL?

- A scripting language
- Designed to abstract game constructs
- Flexible
- Integrated with Java/Scala Swing

What is GameL?

Abstrations:

- Entities

What is GameL?

Abstrations:

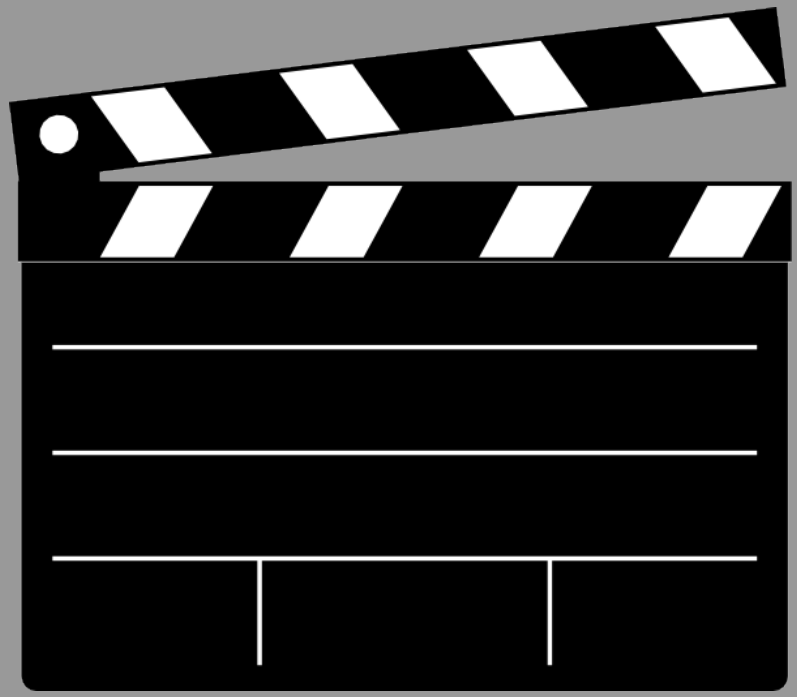
- Entities
- Ownership

http://forums.ubi.com/archive/index.php/t-775003.html

What is GameL?

Abstrations:

- Entities
- Ownership
- Actions



http://properpropaganda.net/2013/08/4-tips-on-designing-calls-to-action-ctas/

What is GameL?

Abstrations:

- Entities
- Ownership
- Actions
- Scenes

http://upload.wikimedia.org/wikipedia/commons/5/58/Sunset_2007-1.jpg

What is GameL?

Abstrations:

- Entities
- Ownership
- Actions
- Scenes
- Transitions

http://www.peninsulabikeparty.org/wp-content/uploads/2013/02/Full-moon-over-water.jpg

Entities and Instances

Entities

- represent types of things in the game

Instances

Entities and Instances

Entities

- represent types of things in the game
- can be instantiated

Instances

Entities and Instances

Entities

- represent types of things in the game
- can be instantiated

Instances

- are Instances of Entities

Entities and Instances

Entities

- represent types of things in the game
- can be instantiated

Instances

- are Instances of Entities
- are the actual things in the game

Entities and Instances

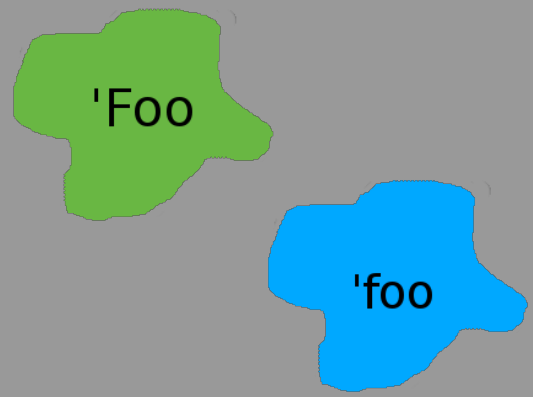
```
define a new entity {
    name = 'Foo
}

create a new instance {
    name = 'foo
} of 'Foo
```

Entities and Instances

```
define a new entity {
    name = 'Foo
}

create a new instance {
    name = 'foo
} of 'Foo
```



Entities and Instances

Entities and Instances can also have attributes:

```
define a new entity {
    name = 'Foo
    attributes += (
        "description" -> "very foo-ish",
        "color" -> 0xa5a5a5,
        "position" -> (0, 0)
    )
}
create a new instance {
    name = 'foo
    attributes += (
        "restaurant" -> "La Bureau de Beurre",
        "pet" -> new Dog("Doggie")
    )
} of 'Foo
```

Entities and Instances

To get the value of an attribute:

```
var food = 'foo tell "restaurant"
var pos  = 'foo tell "position"
```

Ownership

- Entities can own other Entities

Ownership

- Entities can own other Entities
- No multiple ownership

Ownership

- Entities can own other Entities
- No multiple ownership
- `nobody` owns the leftovers

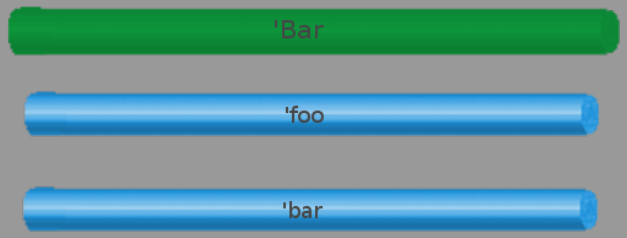
Ownership

- Entities can own other Entities
- No multiple ownership
- `nobody` owns the leftovers
- Ownership can be transfered

Ownership

Entity vs Instance

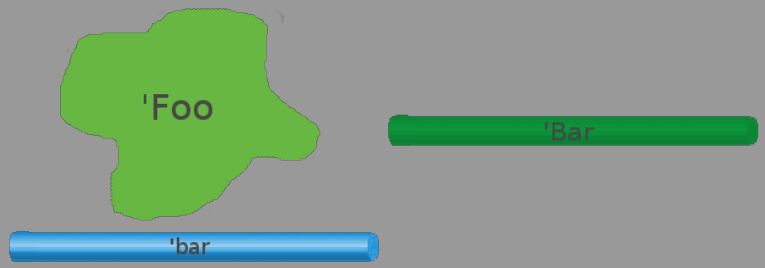
```
define a new entity {
    name = 'Bar
}
create a new instance {
    name = 'bar
} of 'Bar
create a new instance {
    name = 'foo
    objects += (
        'bar
    )
} of 'Bar
```



Ownership

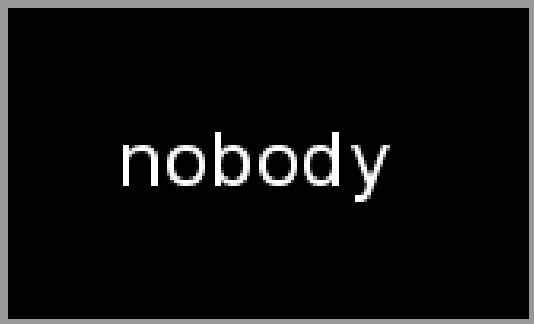
Entity vs Instance

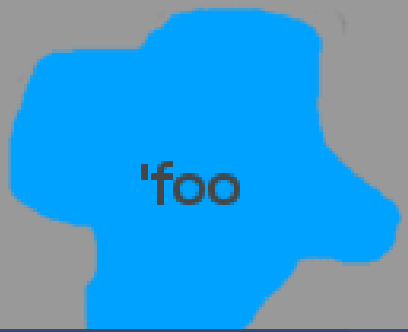
```
define a new entity {
    name = 'Bar
}
create a new instance {
    name = 'bar
} of 'Bar
define a new entity {
    name = 'Foo
    objects += (
        'bar
    )
}
```



Ownership

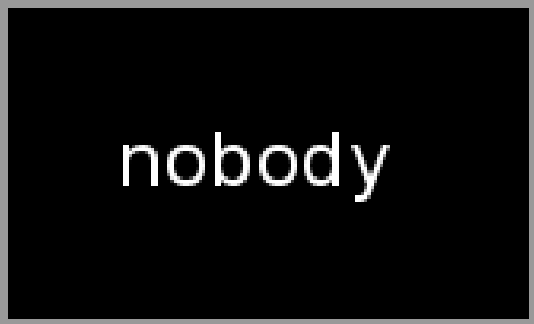
```
nobody gives 'foo to 'bar
// nobody gave 'foo to 'bar
```





Ownership

```
nobody gives 'foo to 'bar
// nobody gave 'foo to 'bar
```



Ownership

```
'bar gives 'foo to 'baz
// 'bar gave 'foo to 'baz
```

Ownership

```
'bar gives 'foo to 'baz
// 'bar gave 'foo to 'baz
```

Ownership

To check ownership:

```
if('foo has 'bar)
```

To get a list of owned instances:

```
'foo.inventory
```

Actions

- User-defined responses to actions

Actions

- User-defined responses to actions
- Manual and automatic triggering

Actions

- User-defined responses to actions
- Manual and automatic triggering
- Reusable

Actions

```
define a new action {
    name = 'fooing
    action = (args: List[Any]) =>
        println("Fooooo!");
}
```

Actions

```
define a new action {
    name = 'fooing
    action = (args: List[Any]) =>
        println("Fooooo!");
    condition = () => amIReady()
}
```

Actions

```
define a new entity {
    name = 'Foo
    actions += (
        'fooing
    )
}
create a new instance {
    name = 'foo
}
```

Actions

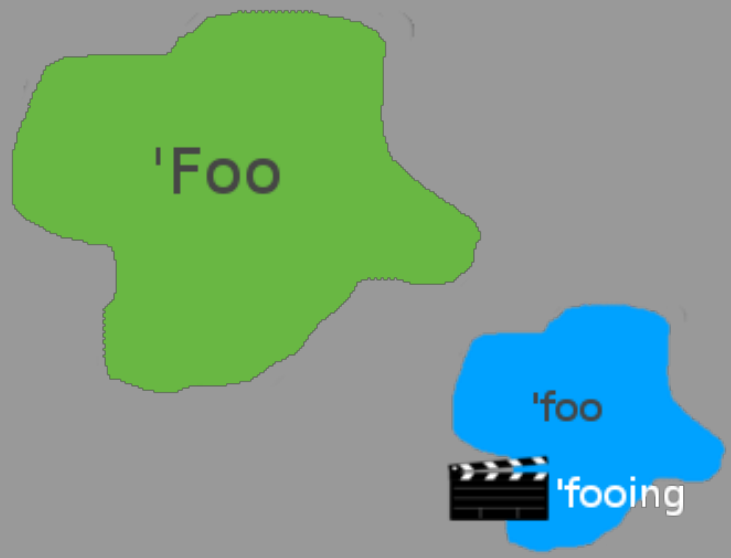
```
'foo does 'fooing using ()
```

Actions

```
'foo does 'fooing using ()
```

Actions

```
define a new entity {
    name = 'Foo
}
create a new instance
    {
    name = 'foo
    actions += (
        'fooing
    )
} of 'Foo
```



Actions

```
'foo does 'fooing using ()
```

Actions

```
'foo does 'fooing using ()
```

Scenes

- represent locations in the game

Scenes

- represent locations in the game
- can own Entities

Scenes

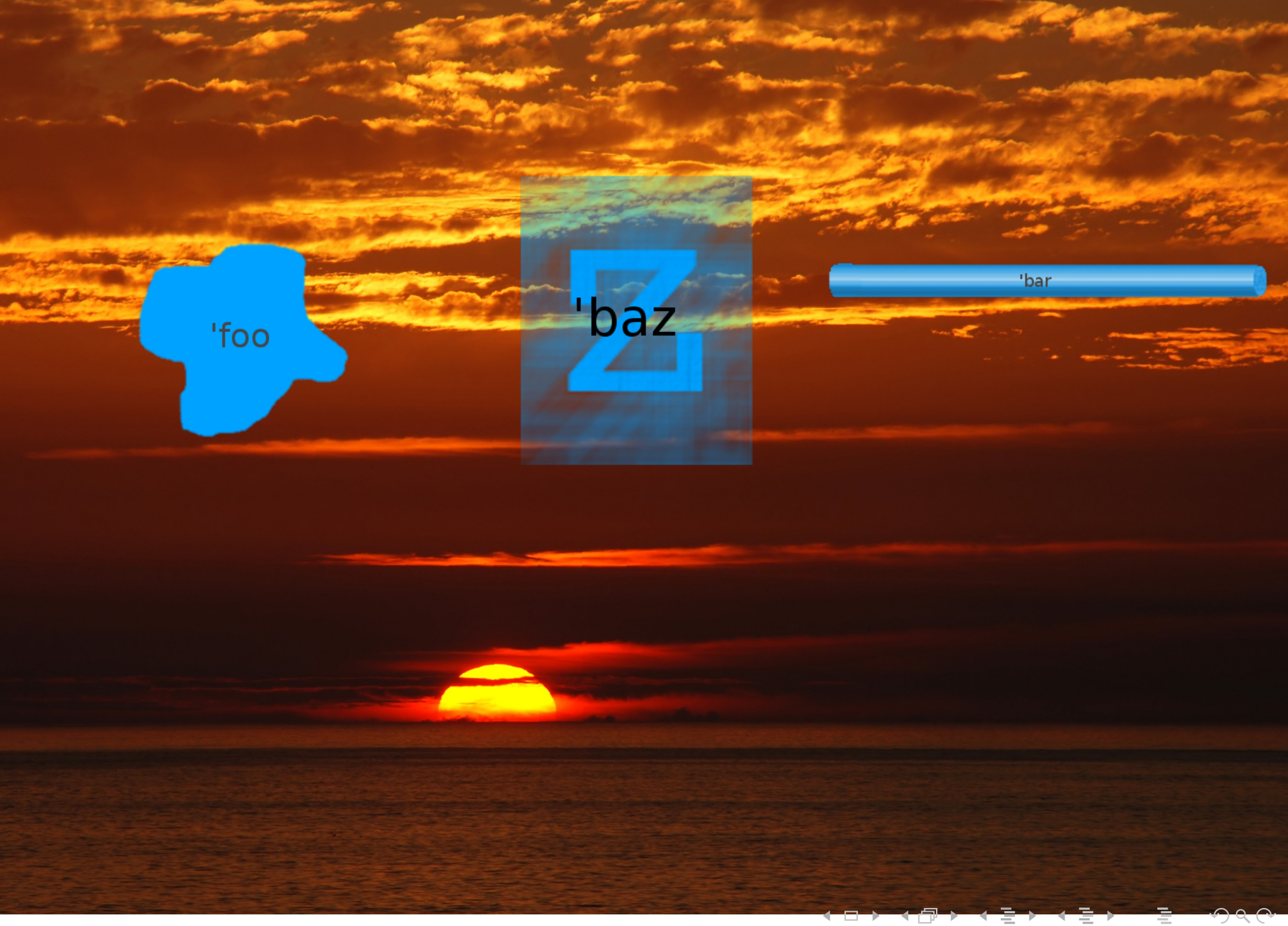
- represent locations in the game
- can own Entities
- can have actions

Scenes

```
create a new scene {
    name = 'Place1
    objects += (
        'foo,
        'baz
    )
    actions += (
        'fooing,
        'bazing
    )
}
nobody gives 'bar to 'Place1
'Place1 gives 'bar to 'foo
'foo gives 'bar to 'Place1
'Place1 does 'bazing using (10, 11, 12)
```


'foo
'baz
'bar


'foo
'baz
'bar


'foo
'baz
'bar


'foo
'baz
'bar
Bazzzz!!

Game

The game object contains the settings of the game

```
create a new game {
    name = "Hello World! Adventure"
    description = "Hello World!"
    resolution = (1024, 512)
    startScene = 'StartScene
    fps = 60 // optional, default=30
}
```

Graphics

- Integrated with Swing

Graphics

- Integrated with Swing
- Automatic rendering

Graphics

- Integrated with Swing
- Automatic rendering
- Support for backgrounds

Graphics

GamelRenderer

- Encapsulates rendering info and methods

Graphics

GamelRenderer

- Encapsulates rendering info and methods
- Can use with scenes or entities

Graphics

GamelRenderer

Just subclass GamelRenderer:

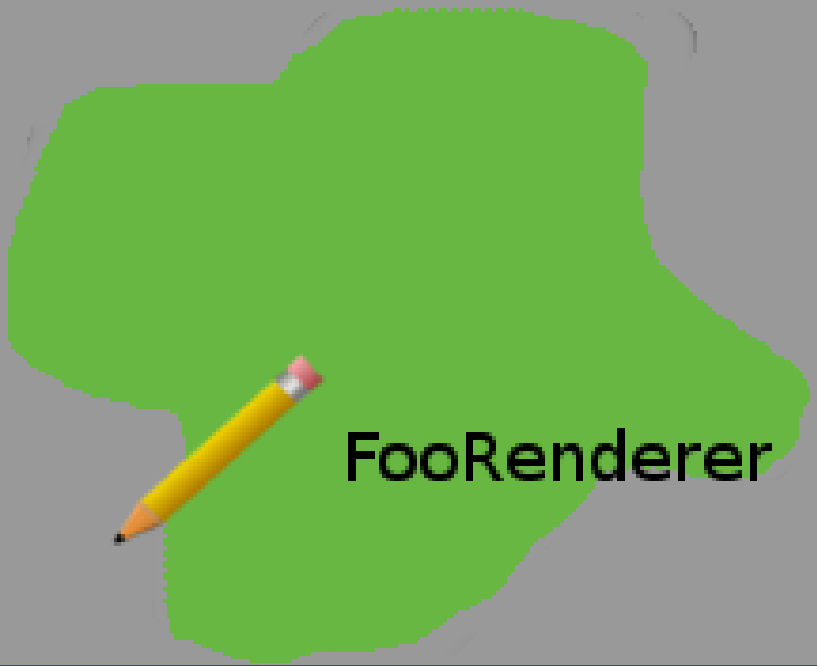
```
object SceneRenderer extends GamelRenderer {
    def render(self: GamelEntity,
               g2d: Graphics2D): Unit = <function>
}
```

Graphics

GamelRenderer

Pass the GamelRenderer to the scene, entity, or instance:

```
define a new entity {
    name = 'Foo
    renderer = FooRenderer
}
```



Graphics

GamelRenderer

Pass the GamelRenderer to the scene, entity, or instance:

```
create a new instance {
    name = 'bar
    renderer = BarRenderer
} of 'Bar
```

Graphics

GamelRenderer

Pass the GamelRenderer to the scene, entity, or instance:

```
create a new scene {
    name = 'Sunset
    renderer = SceneRenderer
}
```

Graphics

Rendering

- All of the current scene's objects are rendered

Graphics

Rendering

- All of the current scene's objects are rendered
- If an entity is rendered, all of its objects are too

Graphics

Rendering

- All of the current scene's objects are rendered
- If an entity is rendered, all of its objects are too
- If an entity or scene does not have a renderer, it is not rendered

Graphics

Switching scenes

```
go to 'Moonrise
```

Graphics

Switching scenes

```
go to 'Moonrise
```

Graphics

Require and Use

- Support for more easily using images in game design

Graphics

Require and Use

- Support for more easily using images in game design
- Allows "aliasing" of images

Graphics

Require and Use

- Support for more easily using images in game design
- Allows "aliasing" of images
- The GamelRenderer also has a `scene` field for backgrounds

Graphics

Require and Use

```
require image "sunset.jpg" as "bg"
// now sunset.jpg can be refered to as "bg"

...

object SceneRenderer extends GamelRenderer {
   var scene = use image "bg"
   // using image as the bg
   def render...
}
```

Event handling

Support for many keyboard and mouse events:

- Key press and release

Event handling

Support for many keyboard and mouse events:

- Key press and release
- Typing

Event handling

Support for many keyboard and mouse events:

- Key press and release
- Typing
- Clicking

Event handling

Support for many keyboard and mouse events:

- Key press and release
- Typing
- Clicking
- Mouse over

Event handling

Support for many keyboard and mouse events:

- Key press and release
- Typing
- Clicking
- Mouse over
- Mouse moving

Event handling

Support for many keyboard and mouse events:

- Key press and release
- Typing
- Clicking
- Mouse over
- Mouse moving
- Mouse press and release

Event handling

Support for many keyboard and mouse events:

- Key press and release
- Typing
- Clicking
- Mouse over
- Mouse moving
- Mouse press and release
- Dragging

Event handling

Support for many keyboard and mouse events:

- Key press and release
- Typing
- Clicking
- Mouse over
- Mouse moving
- Mouse press and release
- Dragging
- Scrolling

Event handling

To toggle an event handler:

```
turn KeyPress on
turn MousePress off
...
```

To detect an event:

```
detect KeyPress
```

Starting the game

```
start game
```

Advantages of GameL

- Modular

Advantages of GameL

- Modular
- Easy to learn

Advantages of GameL

- Modular
- Easy to learn
- Easy to read

Advantages of GameL

- Modular
- Easy to learn
- Easy to read
- Easy to use

Advantages of GameL

- Modular
- Easy to learn
- Easy to read
- Easy to use
- Lazy evaluation => no forward declaration

Advantages of GameL

- Modular
- Easy to learn
- Easy to read
- Easy to use
- Lazy evaluation => no forward declaration
- Flexible

DEMO