

DungeonBots



Team Undead Pixels

Stewart Charles
Wesley Oates
Kevin Parker
Ken Richard



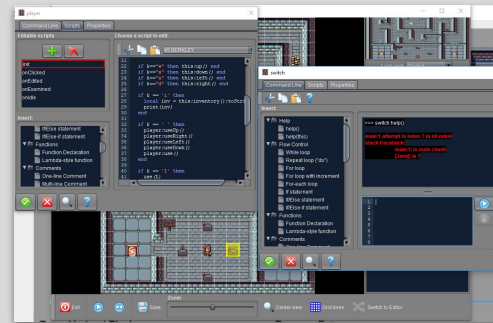
Overview

DungeonBots is the coding education game

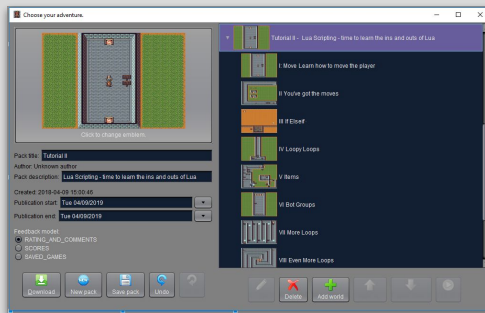
- Fills multiple functions
- Graphically stylish and interesting
- A curriculum delivery vehicle



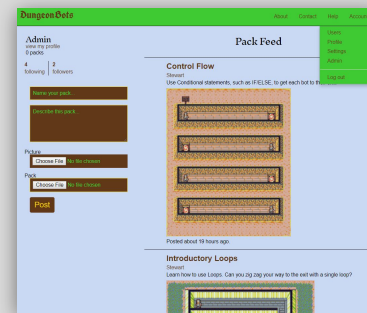
A game and game engine



An IDE



Lesson plans



A community

Is it a level? Is it a lesson?

System Design

Game

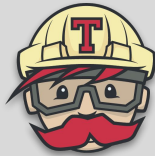
Created our own game custom game engine with Java 8

- Advanced Lua scripting integrations using Java reflection and annotations
- Highly concurrent architecture

Created our own Level Editor from scratch

Website

Back end	Ruby-on-Rails
Front end	Bootstrap, SASS
Database	PostgreSQL
Development	Cloud9
Deployment	Heroku, AWS S3
API	custom



PostgreSQL



Bootstrap



System Capabilities - Game

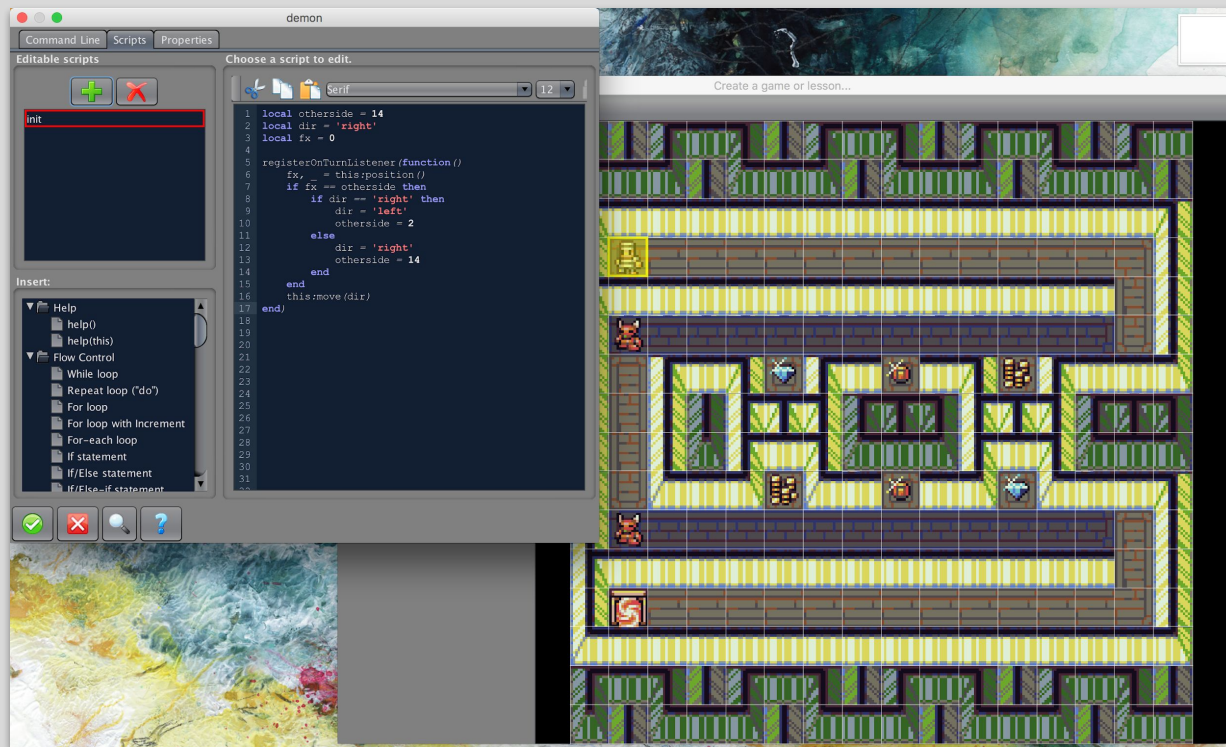
GUI level editor

Lua script editor

Lua grammar / syntax tool

Download levels from website in client

Author-defined help



System Capabilities - Website

Upload & download Level Packs

Pack Feed of user content

Help content for in-game commands

Normal user features, security

Leave us your feedback!

The screenshot displays the DungeonBots website interface. At the top, there is a navigation bar with 'About', 'Contact', 'Help', and 'Account' links. The main content area is divided into several sections:

- Admin:** A user profile for 'Stewart' with 2 packs, 4 following, and 2 followers. It includes form fields for 'Name your pack...', 'Describe this pack...', 'Picture', and 'Pack', each with a 'Choose File' button and 'No file chosen' text. A 'Post' button is at the bottom.
- Pack Feed:** A list of user-generated content. The first entry is 'Control Flow' by Stewart, described as 'Use Conditional statements, such as IF/ELSE, to get each bot to the...'. It features a preview image of a game level with a character and a red 'S' icon. Below the image, it says 'Posted about 19 hours ago.' The second entry is 'Introductory Loops' by Stewart, described as 'Learn how to use Loops. Can you zig zag your way to the exit with a single loop?'. It also has a preview image of a game level.
- Help:** A section titled 'Treasure' with a sub-header 'DungeonBots'. It contains a table of methods with columns for Name, Role, and Description. The methods listed are 'getName' (Role: NONE), 'setName' (Role: AUTHOR), 'getid' (Role: AUTHOR), 'new' (Role: DEFAULT, Description: Create), 'getDescription' (Role: AUTHOR, Description: Gets a...), and 'getResponseQuestions' (Role: AUTHOR, Description: Get the...).

System Limitations

- Supports deletion of entities, but no built-in combat system
- Large built-in library of tiles and entities, but doesn't support custom assets
- Has GUI for generating code, but no autocompletion
- Pack size limitations on website

Unique Features

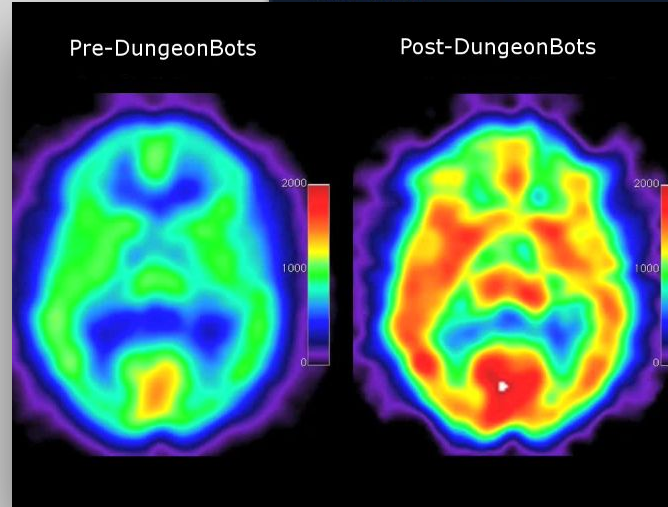
There's nothing like this out there on the market

Learn to program by playing a game

Sandboxed coding environment per entity - every entity can have its own "brain"

Endlessly customizable

```
function PriorityQueue.push(item)
  local idx = #pq+1
  pq[idx] = item
  --Percolate up
  local parentIdx = intDivideBy2(idx)
  while parentIdx >= 1 do
    --player:say("idx:" .. idx .. " parentIdx:" .. parentIdx)
    if pq[parentIdx].priority <= pq[idx].priority then break end
    pq[idx] = pq[parentIdx]
    pq[parentIdx] = item
    idx = parentIdx
    parentIdx = intDivideBy2(idx)
  end
end
function PriorityQueue.pop()
  local result = pq[1]
  pq[1] = pq[#pq]
  pq[#pq] = nil
  --Percolate down
  local idx = 1
  while true do
```



```
    if pq[childIdx].priority > pq[idx].priority then break end
    pq[idx] = pq[childIdx]
    pq[childIdx] = item
    idx = childIdx
  end
  return result
end
```

System Demo

