Visual Abstractions

for Framework, Tool and Language Design

Anyone who only speaks English?

Kick-off, Today

- “Visual Abstractions”
- Topic Suggestions, Bring Your Own Topic
- Organization of Seminar

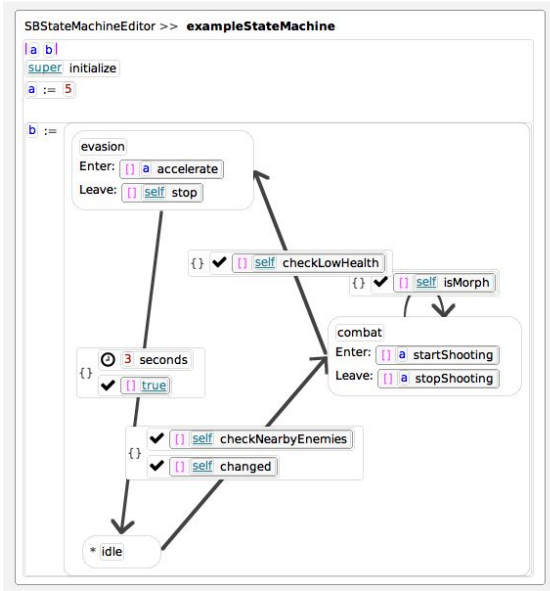
Visual Abstractions

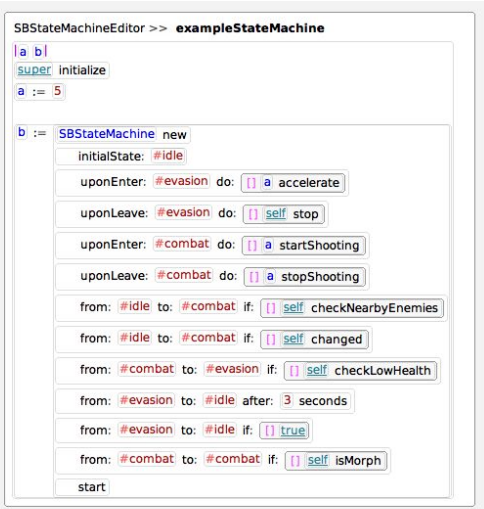
- Abstractions
  - Hide details to reduce mental load
  - Can apply to data, functions, syntax, …

```
def getContentsOfFile(filename):
     stream = open(filename, “r”)
     buffer = []
     while ((data = read(stream)) != EOF):
          buffer.add(data)
     stream.close()
     return buffer
```

Visual Abstractions

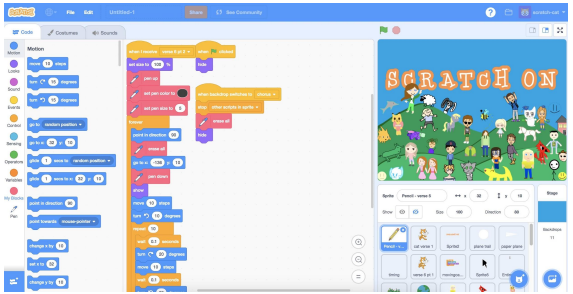
- Abstractions
  - Hide details to reduce mental load
  - Can apply to data, functions, syntax, …
- Visual Abstractions
  - Abstract on the user interface level
  - Bring tools to their context



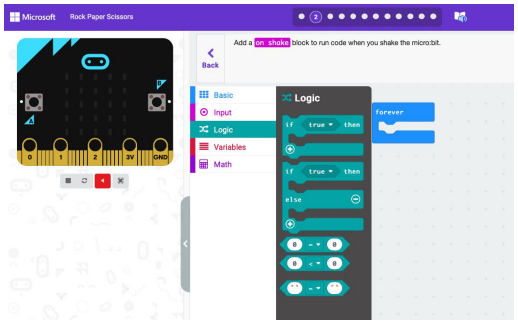


Visual Abstractions

- Abstractions
  - Hide details to reduce mental load
  - Can apply to data, functions, syntax, …
- Visual Abstractions
  - Abstract on the user interface level
  - Bring tools to their context
- Expected to be particularly helpful for...
  - Programming novices
  - Code we rarely touch (config files, utilities, …)
  - Eventually: general-purpose programming



SCRATCH

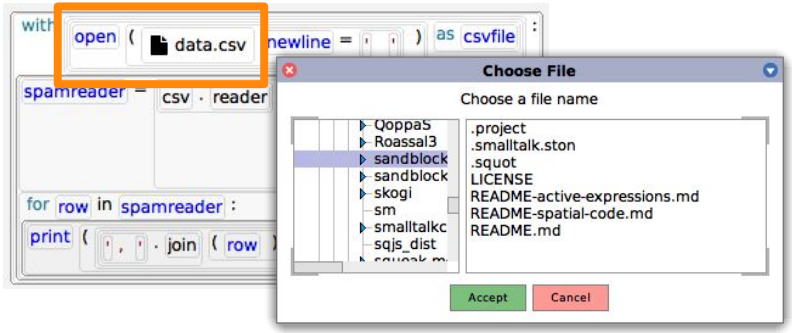


Microsoft MakeCode

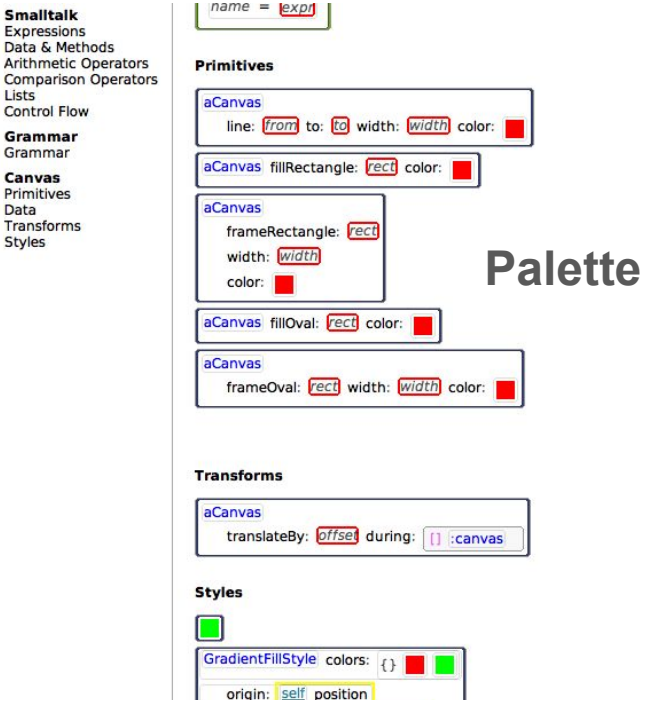
```
drawOn: aCanvas
      aCanvas fillRectangle: ___ fillStyle: Color green
```

“Ingredients”

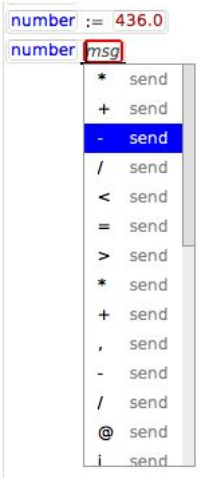
Visual Abstractions

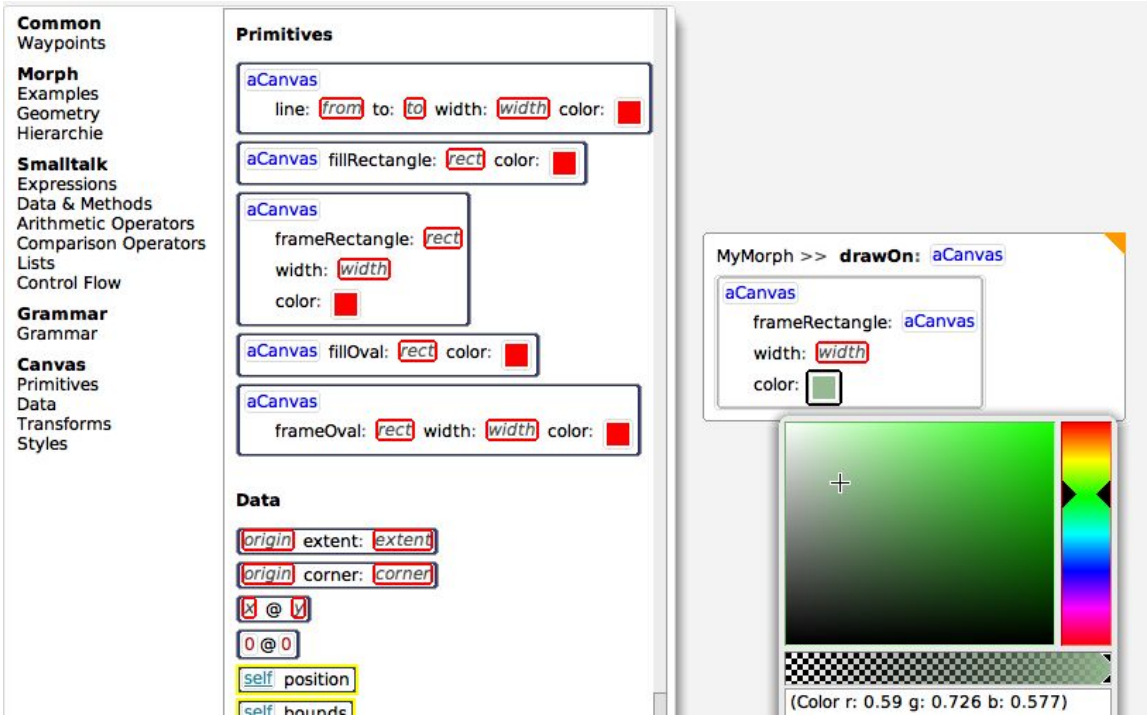


Palette



Syntax-directed Editing and Semantic Autocompletion




Common
Waypoints
Morph
Examples
Geometry
Hierarchie
Smalltalk
Expressions
Data & Methods
Arithmetic Operators
Comparison Operators
Lists
Control Flow
Grammar
Grammar
Canvas
Primitives
Data
Transforms
Styles
Primitives
aCanvas line: from to: to width: width color:
aCanvas fillRectangle: rect color:
aCanvas frameRectangle: rect width: width color:
aCanvas fillOval: rect color:
aCanvas frameOval: rect width: width color:
Data
origin extent: extent
origin corner: corner
x @ y
0 @ 0
self position
MyMorph >> drawOn: aCanvas
aCanvas frameRectangle: aCanvas width: width color:
(Color r: 0.59 g: 0.726 b: 0.577)

Example Workflow

- Take piece of desired source code
- Identify difficult-to-use/non-discoverable aspects of used APIs
- Replace identified aspects with visual abstractions that generate underlying code / introduce palette elements / provide autocompletion

- Implementation platform: Sandblocks
  - Environment for code editing supporting fine-grained mix of code and visual elements
  - Note: research prototype, errors are expected. Of course, you won't have to fix them, instead they will be fixed for you, but expect to have to carry through some workarounds :)

Over The Course of The Semester…

- Alternating every week:
  - “Stuhlkreis”: each person gets 1-5 min to show their progress (Hybrid/Online)
  - Meet with supervisor (In-person/Online)

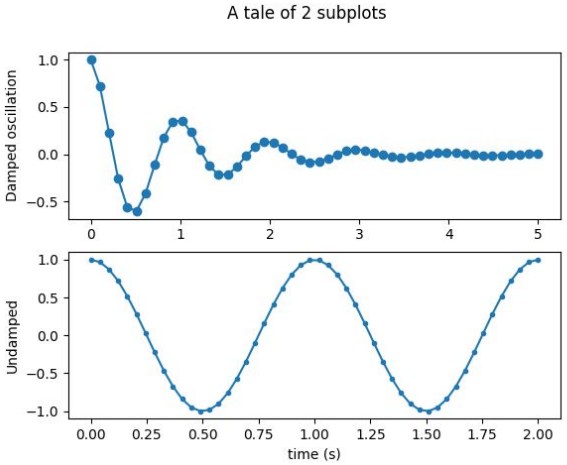
- Ideally: everyone works on the same branch, so we can try out everyone’s demos immediately

- February: Final presentation (rationale, approach, demo)

Topic Suggestions

Chart Configuration (e.g. Python/R/…)

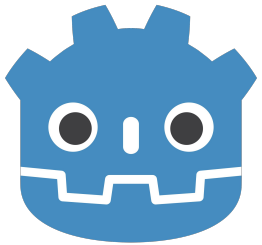
- Usually either entirely visual (Excel, …) or entirely in code (matplotlib, D3, …)
- Goal: enable **discoverable** configuration of graphs that still allows using code where appropriate (loop for subfigures, variables, …)

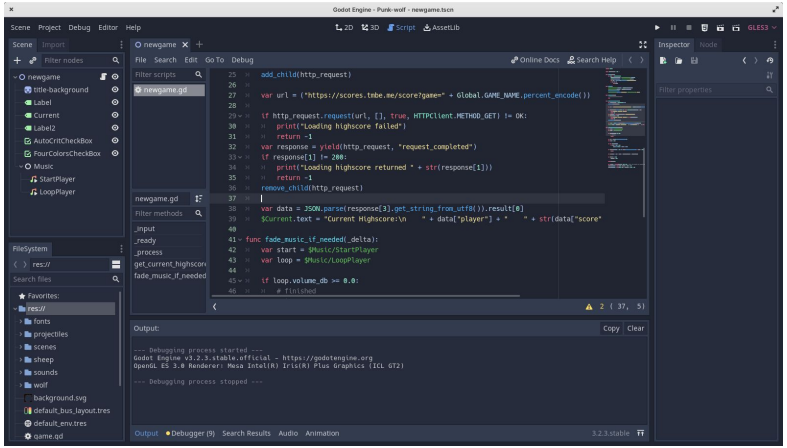


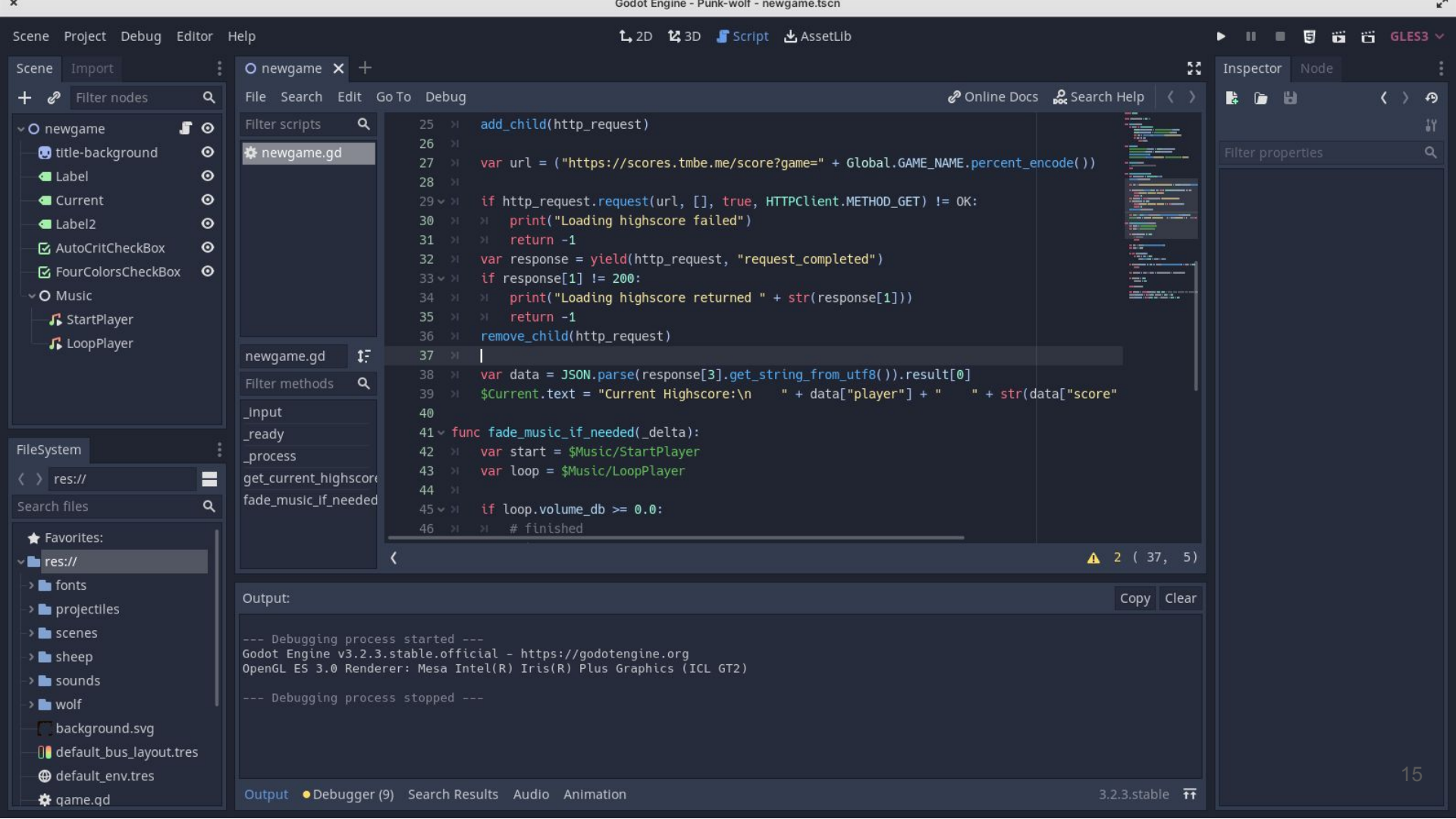

```
import numpy as np
import matplotlib.pyplot as plt

x1 = np.linspace(0.0, 5.0)
x2 = np.linspace(0.0, 2.0)

y1 = np.cos(2 * np.pi * x1) * np.exp(-x1)
y2 = np.cos(2 * np.pi * x2)

fig, (ax1, ax2) = plt.subplots(2, 1)
fig.suptitle('A tale of 2 subplots')

ax1.plot(x1, y1, 'o-')
ax1.set_ylabel('Damped oscillation')

ax2.plot(x2, y2, '.-')
ax2.set_xlabel('time (s)')
ax2.set_ylabel('Undamped')

plt.show()
```

Game Engine Integration (e.g., GDScript, C#, …)

- Example of GDScript, language used in Godot Game Engine
- Various difficult-to-recall interfaces (web requests, multiplayer, rigid body physics, signals, …)
- Classes have “odd” structure (getters, setters, fields, extends)

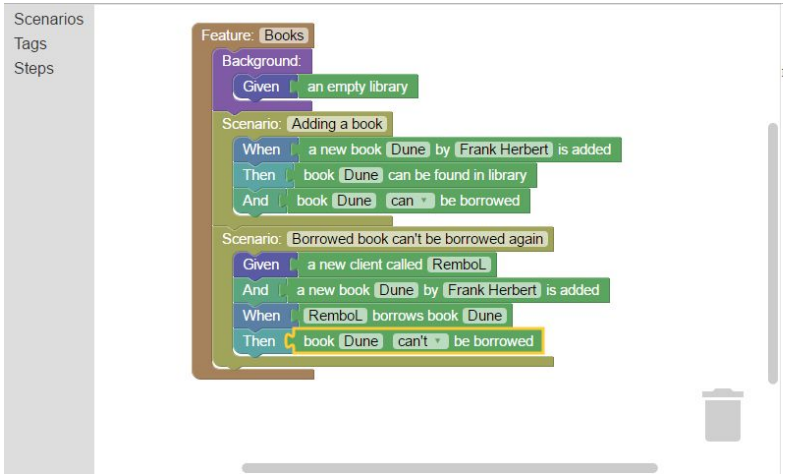




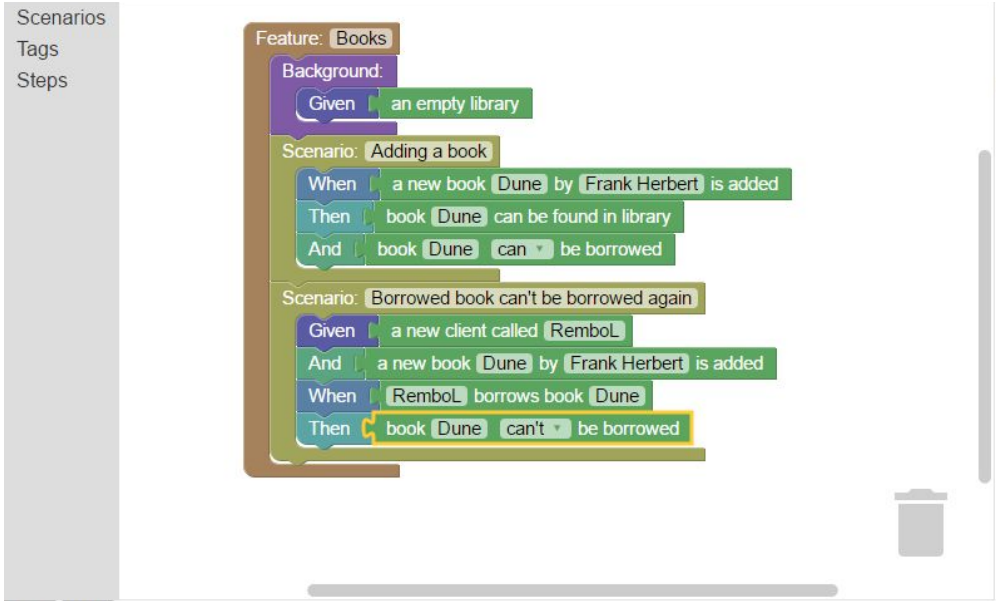

Godot Engine - Punk-wolf - newgame.tscn
Scene Project Debug Editor Help
2D 3D Script AssetLib
GLES3
Scene Import
Filter nodes
newgame
title-background
Label
Current
Label2
AutoCritCheckBox
FourColorsCheckBox
Music
StartPlayer
LoopPlayer
FileSystem
res://
Search files
Favorites:
res://
fonts
projectiles
scenes
sheep
sounds
wolf
background.svg
default_bus_layout.tres
default_env.tres
game.gd
newgame
File Search Edit Go To Debug
Online Docs Search Help
Filter scripts
newgame.gd
newgame.gd
Filter methods
_input
_ready
_process
get_current_highscor
fade_music_if_needed
add_child(http_request)
var url = ("https://scores.tmbe.me/score?game=" + Global.GAME_NAME.percent_encode())
if http_request.request(url, [], true, HTTPClient.METHOD_GET) != OK:
print("Loading highscore failed")
return -1
var response = yield(http_request, "request_completed")
if response[1] != 200:
print("Loading highscore returned " + str(response[1]))
return -1
remove_child(http_request)
var data = JSON.parse(response[3].get_string_from_utf8()).result[0]
$Current.text = "Current Highscore:\n " + data["player"] + " " + str(data["score"
func fade_music_if_needed(_delta):
var start = $Music/StartPlayer
var loop = $Music/LoopPlayer
if loop.volume_db >= 0.0:
# finished
2 (37, 5)
Output:
Copy Clear
--- Debugging process started ---
Godot Engine v3.2.3.stable.official - https://godotengine.org
OpenGL ES 3.0 Renderer: Mesa Intel(R) Iris(R) Plus Graphics (ICL GT2)
--- Debugging process stopped ---
Output Debugger (9) Search Results Audio Animation
3.2.3.stable
Inspector Node
Filter properties
15

Cucumber BDD

- Behavior driven tests aim to bridge the gap to the programming domain
- Goal: Explore means to work with variables, templates, data, … in a visual manner rather than text to reduce likelihood of incompatibilities or misunderstandings



https://github.com/RemboL/blockumber




```
Feature: Books

  Background:              # test.feature:3
    Given an empty library # LibrarianStepdefs.groovy:28

  Scenario: Adding a book                          # test.feature:6
    When a new book Dune by Frank Herbert is added # LibrarianStepdefs.groovy:34
    Then book Dune can be found in library         # LibrarianStepdefs.groovy:38
    And book Dune can be borrowed                  # LibrarianStepdefs.groovy:42

  Background:              # test.feature:3
    Given an empty library # LibrarianStepdefs.groovy:28

  Scenario: Borrowed book can't be borrowed again # test.feature:11
    Given a new client called RemboL              # LibrarianStepdefs.groovy:20
    And a new book Dune by Frank Herbert is added # LibrarianStepdefs.groovy:34
    When RemboL borrows book Dune                 # LibrarianStepdefs.groovy:46
    Then book Dune can't be borrowed              # LibrarianStepdefs.groovy:42
```

```ruby
Given "$name has a valid Credit or Debit card" do |name|
  # Ruby code
end
```

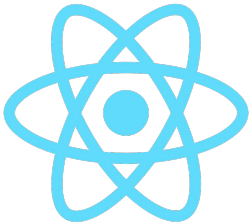
Web Server (e.g. JavaScript, Ruby, …)

- Overview for middleware
- Integration with runtime to show recent requests
- Mixing with other languages, e.g. templating or SQL

```ruby
require 'sinatra/base'

class LoginScreen < Sinatra::Base
  enable :sessions

  get('/login') { haml :login }

  post('/login') do
    if params['name'] == 'admin' && params['password'] == 'admin'
      session['user_name'] = params['name']
    else
      redirect '/login'
    end
  end
end

class MyApp < Sinatra::Base
  # middleware will run before filters
  use LoginScreen

  before do
    unless session['user_name']
      halt "Access denied, please <a href='/login'>login</a>."
    end
  end

  get('/') { "Hello #{session['user_name']}." }
end
```

React / Angular (JavaScript/Typescript)

- Hooks may be considered somewhat obscure (inline documentation, icons?)
- JSX provides box-like structure that could be enriched by e.g. thumbnails
- Better support for CSS-in-JS



```tsx
import React from "react";
import { CharacterManager } from "./characters/CharacterManager";
import { Collapsible } from "./Collapsible";
import { Resizable } from "re-resizable";
import useLocalState from "../useLocalState";
import { InitiativeTracker } from "./InitiativeTracker";
import { useIsGM } from "../myself";
import { Maps } from "./Maps";
import { Achievements } from "./Achievements";
import { Music } from "./Music";
import { GMArea } from "./GMArea";
import { useServerState } from "../state";
import { DicePanel } from "./DicePanel";

export function Sidebar({ logout }: { logout: () => void }) {
  const [sidebarWidth, setSidebarWidth] = useLocalState("sidebarWidth", 450);
  const myselfIsGM = useIsGM();
  const musicIsGMOnly = useServerState(
    (state) => state.globalSettings.musicIsGMOnly
  );

  return (
    <Resizable
      size={{ width: sidebarWidth, height: "100%" }}
      onResizeStop={(_1, _2, _3, sizeDelta) =>
        setSidebarWidth((width) => width + sizeDelta.width)
      }
      minWidth="10vw"
      maxWidth="90vw"
```

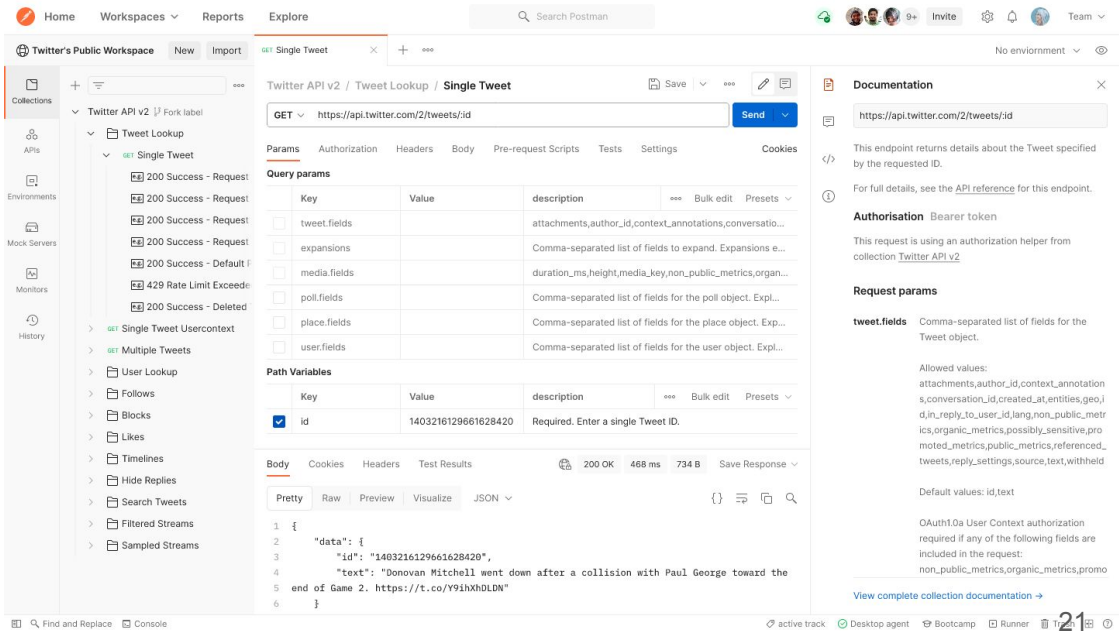
Squeak ToolBuilder (Smalltalk)

- Declarative code, often interleaved with imperative statements
- Goal: inline drag-and-drop UI builder for widgets and their configuration that mixes with imperative statements

```
buildCodePaneWith: builder
    | textSpec top buttonSpec annoSpec |
    self wantsOptionalButtons ifTrue: [
        top := builder pluggablePanelSpec new.
        top children: OrderedCollection new.
        buttonSpec := self buildOptionalButtonsWith: builder.
        buttonSpec frame: self optionalButtonsFrame.
        top children add: buttonSpec].
    textSpec := builder pluggableCodePaneSpec new.
    textSpec
        model: self;
        getText: #contents;
        setText: #contents:notifying:;
        selection: #contentsSelection;
        menu: #codePaneMenu:shifted:.
    self wantsAnnotationPane ifTrue: [
        top ifNil: [
            top := builder pluggablePanelSpec new.
            top children: OrderedCollection new].
        annoSpec := self buildAnnotationPaneWith: builder.
        annoSpec frame: self annotationFrame.
        top children add: annoSpec].
```

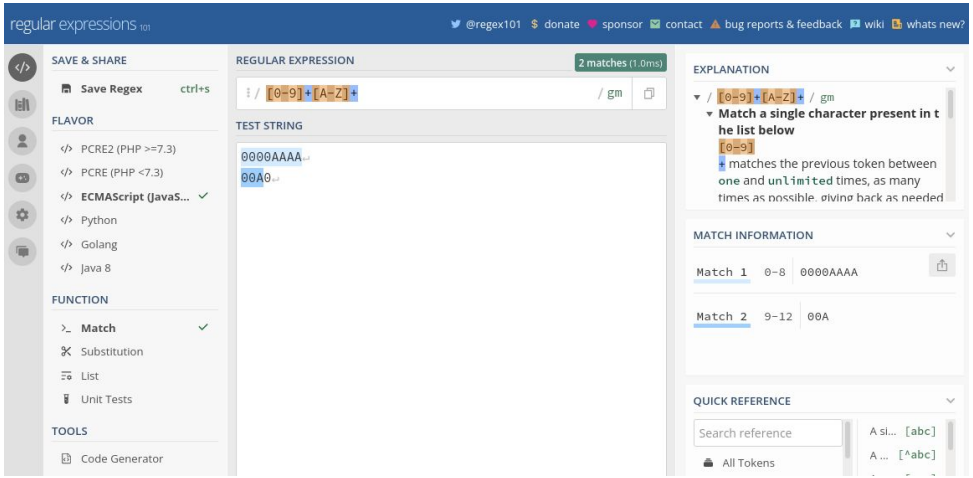
Web Client (e.g. JavaScript, Smalltalk, …)

- “Postman in your code base”
- Easy-to-use configuration and preview of requests
- Embedded examples for each request for debugging etc.



Regex101

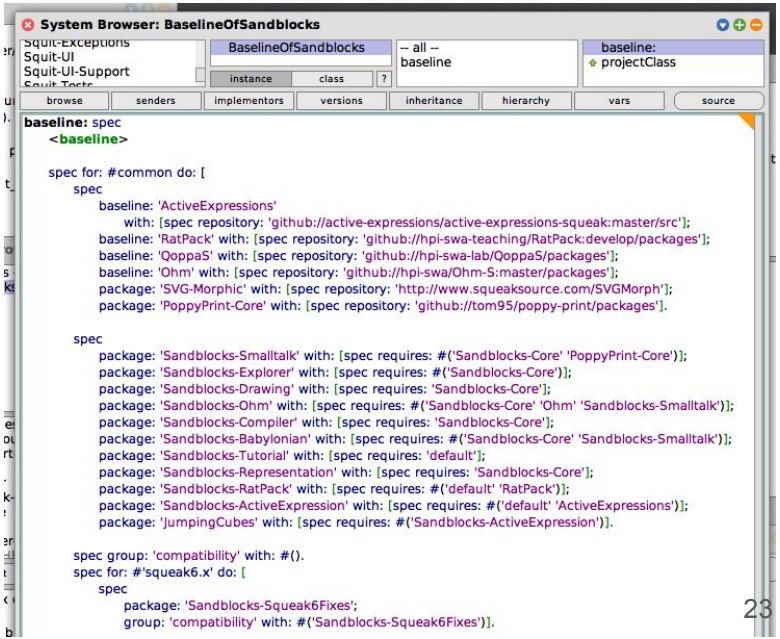
- Provide an inline regex authoring interface similar to regex101
- Examples that pass/fail the regex
- Explanation of regex structure
- Assist in calling the various match/replace API functions



Configuration Languages (YAML/JSON/Smalltalk/…)

- gitlab-ci.yml, Metacello Baselines, .eslintrc.json, …
- Tend to be rarely accessed and have next to no domain-specific tooling
- Provide autocompletion, items in palette

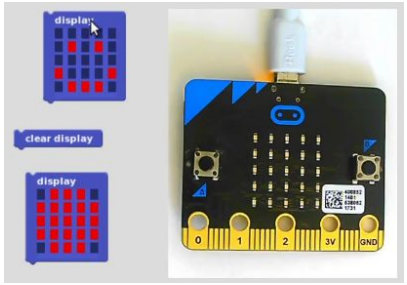
```yaml
name: smalltalkCI

on: push

jobs:
  test:
    runs-on: ubuntu-latest
    strategy:
      matrix:
        smalltalk: [ Squeak64-trunk, Squeak64-5.3 ]
    name: Test ${{ matrix.smalltalk }}
    env:
      GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
    steps:
      - uses: actions/checkout@v2
      - uses: hpi-swa/setup-smalltalkCI@v1
        with:
          smalltalk-image: ${{ matrix.smalltalk }}
      - run: smalltalkci -s ${{ matrix.smalltalk }}
        shell: bash
        timeout-minutes: 15
      - run: ./scripts/build_version.sh
        if: startsWith(github.ref, 'refs/tags/') && matrix.smalltalk == 'Squeak64-5.3'
        shell: bash
```

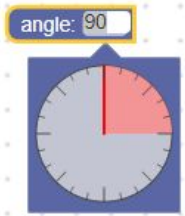


Bring Your Own Topic

- Have a framework/library in mind that you feel would benefit from Visual Abstractions?
- Verify that the language is supported by TreeSitter [1, 2]
- Note 2-3 opportunities where you think Visual Abstractions (VA) are applicable

- For example… My Arduino Project
  - VA for LCD panels
  - VA for angle/number inputs
  - Next steps: generate C code and flash to chip via PlatformIO, read back data





[1] https://tree-sitter.github.io/tree-sitter/#available-parsers
[2] https://tree-sitter.github.io/tree-sitter/playground

Organization

Organization

- Course
  - Project-Seminar, 4 SWS, 1-2 students per group
- Presentations
  - Informal show-of-progress with all participants every two weeks
  - End-term Presentation (Feb)
- Grading
  - 6 ECTS graded credit points
  - Grade based on project and presentation
- Important dates
  - Enrollment with preferred/own topic on or before October, 31st
  - Topic Assignments on November, 2nd
  - Presentation dates determined after topics are assigned

Enrollment

- To enroll please send a mail to tom.beckmann@hpi.de on or before October, 31st, stating...
  - The **framework(s)/librar(ies)** you want to work with
  - The specific **language**
  - If it's your own topic: two bullet points where you see potential for **"Visual Abstractions"**
  - A **backup topic** that you could also imagine working on, in case there is significant overlap

Hand-In Reference Sheet

- Final presentation
  - Presentation file (PDF)
  - Screencast (mp4, public, 1080p)
- Final submission on 18.02. (in zip, link or email)
  - Presentation and screen again, narrated or with subtitles
  - Source code (MIT License)
  - Figure / Screenshot (png)
  - Demo source code snippet
  - Documentation and README.md
- All hand-ins will be archived in a public repository