EECS 498 : Game Engine Architecture

Winter 2024 Mid-Semester Exam

By Austin Yarger - University of Michigan (ayarger@umich.edu)

Topic	Score
Game Engine Landscape, History & Misc	/ 40
C++ and IDE Pragmatics	/ 20
Lua and Composition	/ 20
Engine Architecture and Lifecycle Functions	/ 20
Total	/ 100
Total (canvas normalized)	/ 250

Rookie, we've completed some questions for you.
Be efficient with your time, and good luck out there.

Hey! Do your best with no regrets!
Your path is a difficult one– don't forget that.

Good evening. Before we begin, please repeat after me.
"I have neither given nor received unauthorized aid on this examination, nor have I concealed any violations of the honor code."

name / signature _____

uniqname _____

date _____

Glossary

Helpful Engine Lua API Functions
Audio.Play(int channel, string clip_name, bool does_loop)
Text.Draw(string content, int x, int y, font_name, font_size, int r, int g, int b, int a)
Application.GetFrame() – Returns the current frame as an integer.
Image.DrawUI(string image_name, int x, int y) – Draw an image at (x,y) in screen space.
Input.GetKeyDown(key) – Returns bool if the key was pressed down this frame.
Actor.FindAll(string name) – Returns a table of all actor references with a name.
Actor.Find(string name) – Returns a single actor reference by name.
Debug.Log(message) – Prints a message string to the screen.
Application.Quit() – Quits the application immediately.
actor_ref:GetComponent(ComponentTypeName) – Get reference to component.
Image.Draw(image_name, int x, int y) – Draws image in scene space (x,y) (**note :** no floats)

Helpful Common Lua Functions
math.abs(number) -- Returns absolute value of "number"
table.insert (my_table, thing_to_add) – Adds a thing to a table at the very end.
table.remove(my_table, index_to_remove) -- Removes the item at "index_to_remove"

Helpful Lua Techniques
for index, value in ipairs(table) do <logic_here> end – iterating through a table w/ index.
for i = #my_table, 1, -1 do <logic_here> end – Iterating backwards through a table.
some_number % 2 == 0 – using modulo to check if some number is even.

Helpful C++ / Pseudocode Techniques
LuaRef my_function = component_luaref["my_function"]; // Get a function on lua component
my_function(component_luaref); // call it and send in a reference to "self" (member function).

```
/* Iterate through a map */
std::map<std::string, luabridge::LuaRef> components;
for (auto & pair : components)
{
        std::cout << pair.first << std::endl; // access the string (the key).
        LuaRef actual_component = pair.second; // access the component (the value).
}
```

`this` : a pointer to the current object

Characters


Spot


Donna

Game Engine Landscape, History & Misc

1

__/1

Game engines borrow their terminology from the realm of theater, including "actor" to represent "things", "script" to represent modular, attachable logic, and _____ to represent a collection of actors. **(circle one below)**

scene template book rehearsal package prop

2

__/1

In modern times, two off-the-shelf game engines stand above the rest with significant advantages in popularity, community, and featureset, to the point where many call the situation a "duopoly". Which are they? **(circle two below)**

Source Love2D Unity PyGame Unreal CryEngine

3

__/1

Off-the-shelf engines are extraordinarily popular, but some studios refuse to indulge. Why might a studio use an in-house engine? **(circle two below)**

Avoid royalties Reduced dev time / cost Full agency Large community Easier hiring

4

__/1

OOP-based component architectures are very common in today's most popular engines. Name a significant weakness of this style. **(circle one below)**

Cache Utilization Logic re-use Runtime flexibility

5

__/1

Open-source software is awash with licenses restricting how software may be used. Which one requires you release your source-code? (scary!) **(circle one)**

MIT BSD GPL APACHE

6

__/1

Unity was not the first off-the-shelf engine, but it did change engine access forever. In earlier eras, how did one typically acquire an engine? **(circle two)**

Create it yourself Download for free Massive up-front license fees Minor royalties owed

7

__/1

In the eternal chase for performance, some engines have been forgoing traditional object-oriented programming in favor of **(circle one below)**

Static-Oriented programming Data-Oriented Programming Imperative Programming Declarative Programming

8

__ / 9

Match each term below with its single most-reasonable definition at the bottom of the page. Do so by drawing a line between two boxes. Each term matches exactly one definition. There will be nine lines total.

Sprite

Lua

Collider

Mod

Trigger

b2World

std::shared_ptr

Text Adventure

Konami Code

Too easy…
Let's gooo!

Box2D simulation

Smart Pointer

A scripting language

A visual representation of something

Created to help Q&A teams work faster

A shape enabling physical collision on an actor

A shape enabling collision detection without physical impact

A prominent genre in the early days of computing.

Typically created by a game's community, rather than its original authors.

9

___ / 8

Match each term below with its single most-reasonable definition at the bottom of the page. Do so by drawing a line between two boxes. Each term matches exactly one definition. There will be eight lines total.

Unreal

Godot

PICO-8

MonoGame

Love2D

PyGame

MIT Scratch

Game Maker

Python-based

C# Engine of *Celeste ('18)*

Minimalist 2D with Lua scripting

All games limited to 128x128 pixels

Web-based with "code-block" scripting for education

Industry-grade / Used in film production (*The Mandalorian*)

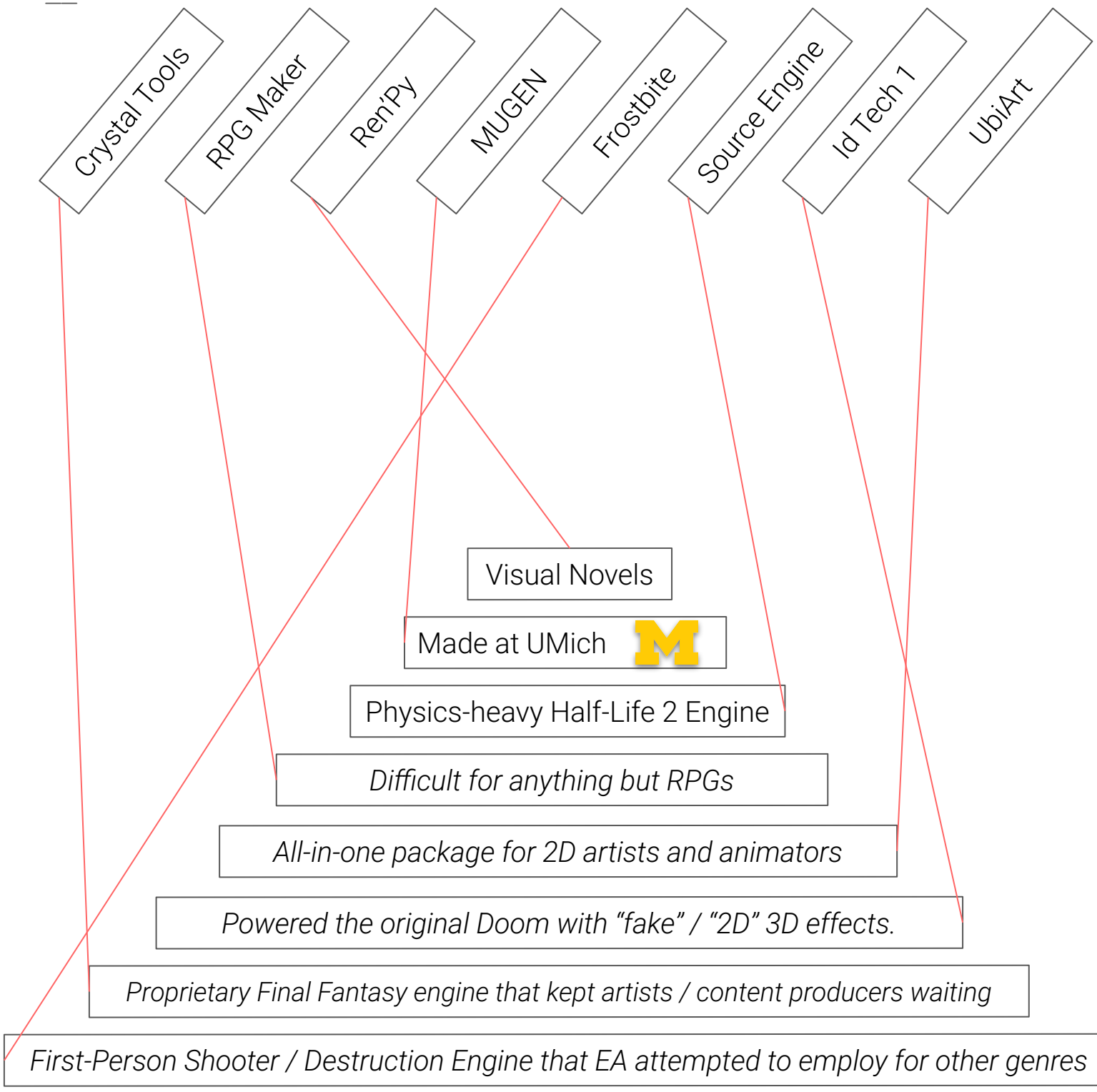
Open-source, MIT-licensed rival to Unity

Drag-and-Drop-based engine of *UnderTale (2015)*

10

___ / 8

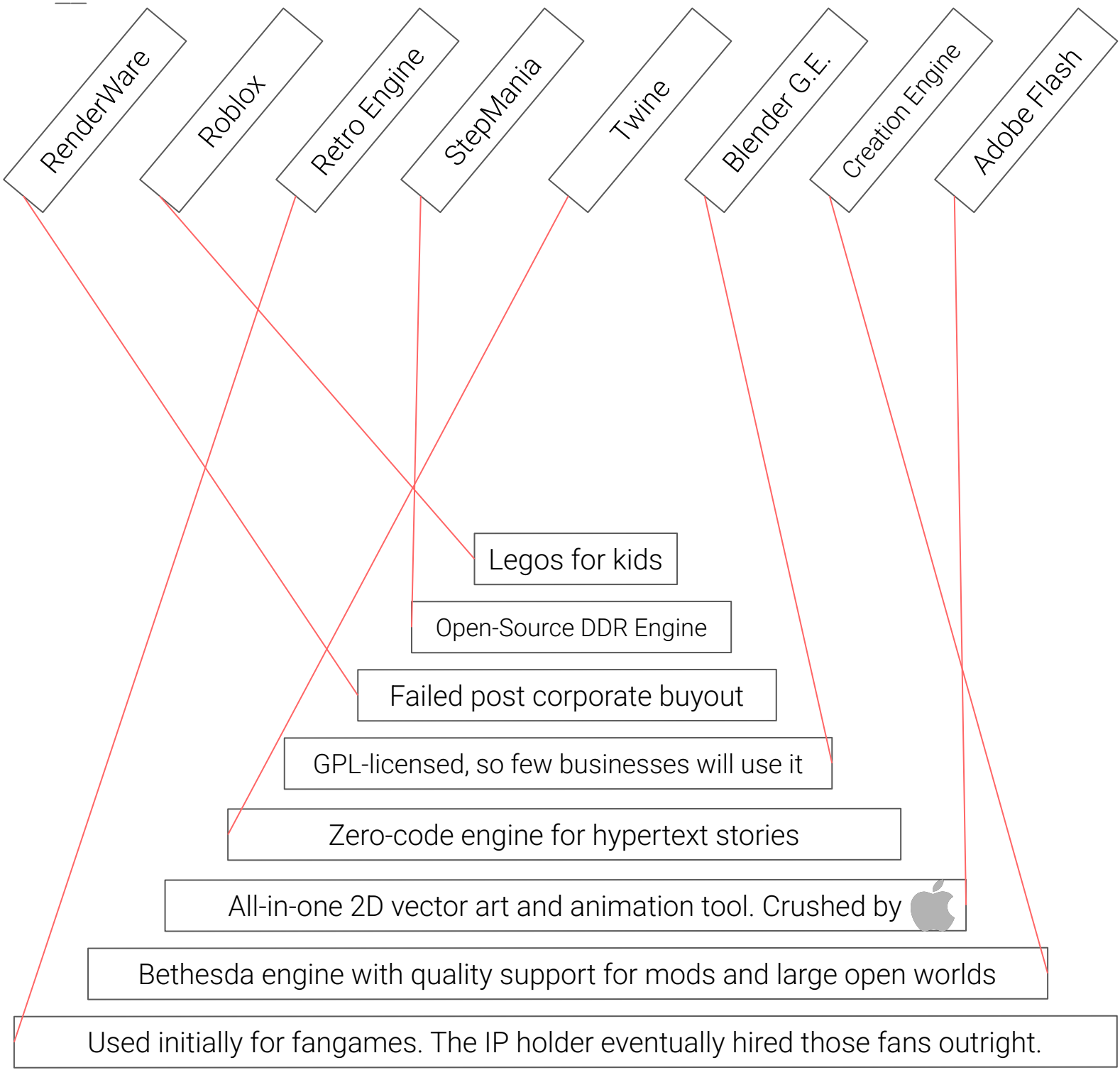
Match each term below with its single most-reasonable definition at the bottom of the page. Do so by drawing a line between two boxes. Each term matches exactly one definition. There will be eight lines total.



11

___ / 8

Match each term below with its single most-reasonable definition at the bottom of the page. Do so by drawing a line between two boxes. Each term matches exactly one definition. There will be eight lines total.



12

__/2

During a build and run of your engine, you witness the following error occur a few moments into the build. What kind of issue is it? **(circle one below)**

```
/usr/bin/ld: /tmp/Rigidbody-b4d7dd.o: in function RigidBody::RigidBody(Actor*)':
Rigidbody.cpp:(.text+0x1c68): undefined reference to b2World::b2World(b2Vec2 const&)' /usr/bin/ld:
Rigidbody.cpp:(.text+0x1c98): undefined reference to b2World::SetContactListener(b2ContactListener*)
```

Compilation issue | Linker issue | Dynamic Linker issue

13

__/2

Your engine begins to run and things look promising until suddenly you witness the following error. What kind of issue is it? **(circle one below)**

```
dyld[63759]: Library not loaded: @rpath/SDL2_ttf.framework/Versions/A/SDL2_ttf Referenced from:
<9A333890-1011-3CE8-BFF6-7880CD371ED2>
```

Compilation issue | Linker issue | Dynamic Linker issue

14

__/2

During a build and run of your engine, an issue appears almost immediately after clicking the build and run button. What kind of issue is it? **(circle one)**

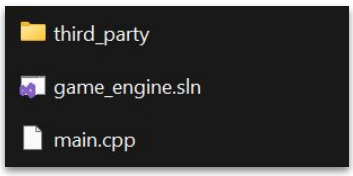
Compilation issue | Linker issue | Dynamic Linker issue

15

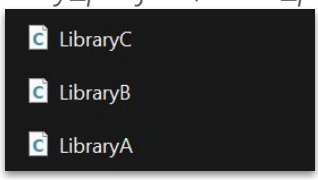
__/4

Consider the following filesystem structure–

my_project/



my_project/third_party/



And the following includes within main.cpp–

```
#include "LibraryA.h"
#include "third_party/LibraryB.h"
#include "third_party/LibraryC.h"

int main()
{
    // ...
    return 0;
}
```

C++ and IDE Pragmatics

Write two include / header paths below that allow for successful compilation without changing the source code of main.cpp. One has been done for you.
Note : You may use VS or XCode syntax.

$(ProjectDir)

$(ProjectDir)third_party

16

__ / 2

You wish to provide engine-level support for tracking of actor-associated network ID ints. Users need fast lookup but no ordering. Which data structure best meets these requirements? **(circle one below)**

std::vector<int> std::map<Actor*, int> std::unordered_map<Actor*, int>

std::unordered_set<int> std::set<int> std::vector<std::pair<Actor*, int>>

17

__ / 6

Consider the following object-oriented c++ script meant to calculate the damage taken by a collection of Characters (only the defense stat is used).

```
struct Character {
    int hp;
    int def;
    int spd;
    int rel;
    int vel;
};

Character characters[]; // "automatically" gets filled with character structs.
int num_chars; // "automatically" gets set to the number of characters.

int main() {

    int damage_amount = 10;
    for(int i = 0; i < num_chars; i++) {
        std::cout << "damage taken : " << (damage_amount - characters[i].def) << std::endl;
    }

    return 0;
}
```

On the next page, re-write this script to speed it up (improve cache utilization).

- Consider changing how Characters are represented in memory, but keep all the stats above (“hp”, “def”, etc must exist in some fashion).
- The program output / calculation should be the same as above.
- If you create an array(s), you may assume it get “filled up” elsewhere.
- You may assume num_chars is the number of entities in the program.

```
int hp[];
int def[];
int spd[];
int rel[];
int vel[];




int num_chars; // "automatically" gets set to the number of characters.

int main()
{
      int damage_amount = 10;
      for (int i = 0; i < num_chars; i++)
      {

            std::cout << "damage taken : " << (damage_amount - def[i]) <<
std::endl;

      }



      return 0;
}
```

18

__/2

A rival to object-oriented programming, the act of focusing on "Objects of Arrays" rather than "Arrays of Objects" (ie, focusing on runtime-efficient data layout) is referred to as **(write below)**–

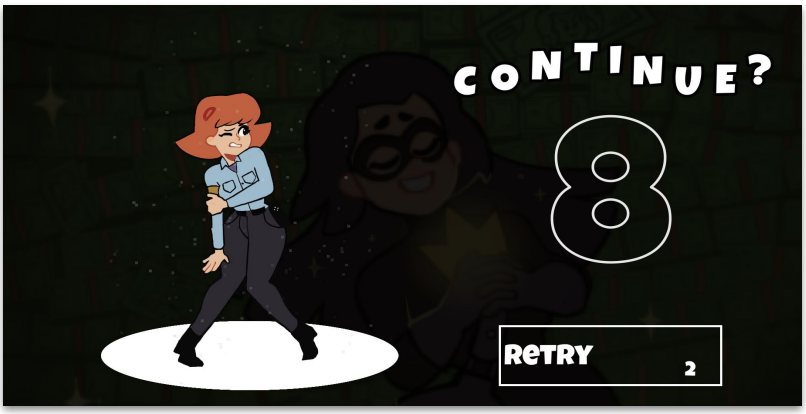
Data-Oriented Programming / Data-Oriented Design

19

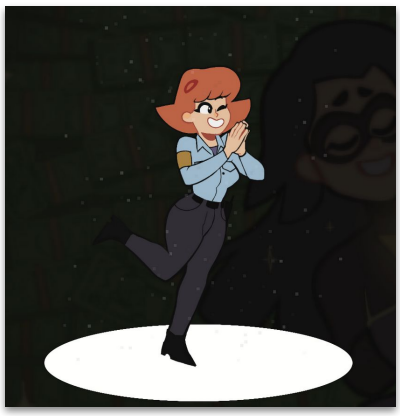
__/10

Quarter Muncher

The business model of 90s-era arcade cabinets incentivized game creators to design difficult gameplay capable of extracting quarters from players every 5 or so minutes. In an effort to coax players into trying again (ie, spend more money), they created dramatic “continue screens” to incentivize continuation.



Example : arcade-style continue countdown



Retry selected

Countdown expired

Objective

Write a Lua component to power a basic arcade-style continue countdown screen.
Note : If a requirement is ambiguous, it is your choice (different solutions may be valid).
Note : Use the glossary at the front of this exam for a reminder of Lua functions / stuff.

Requirements

- ☐ Write your Lua component in the box on the following page.
- ☐ Component must be valid Lua code.
- ☐ Component makes use of several Lua lifecycle functions from lecture / homeworks.
- ☐ Component immediately plays looping audio file called “continue” (continue.wav)
- ☐ Every frame, draw a countdown number text at (500, 100) with font “default” and size 50.
    - ☐ Countdown begins at “9” and decreases by one approximately every 60 frames.
- ☐ Every frame, Image.DrawUI() Spot at (100,100) using pose based on countdown state.
    - ☐ If countdown is > 0, draw “spot_continue” image (her standing pose)
    - ☐ If countdown is <= 0, draw “spot_giveup” image (her sitting pose)
    - ☐ If player has pressed spacebar to continue, draw “spot_retry” image (happy pose)
- ☐ Every frame, draw a “Continue?” text at (500,20) with size 16
- ☐ If player presses “space” before countdown reaches 0, the countdown halts, all text disappears, and spot enters “spot_retry” pose. Component continues on like this forever.
- ☐ If countdown reaches “0”, the countdown halts, all other text disappears, and “GAME OVER!” text renders at (500, 100). Spot renders in her “spot_giveup” pose. It becomes impossible to press “space” to continue. Component continues on like this forever.
- ☐ Component avoids polluting the global lua state, creating local / table variables only.

Lua and Composition

resources/component_types/ContinueCountdown.lua

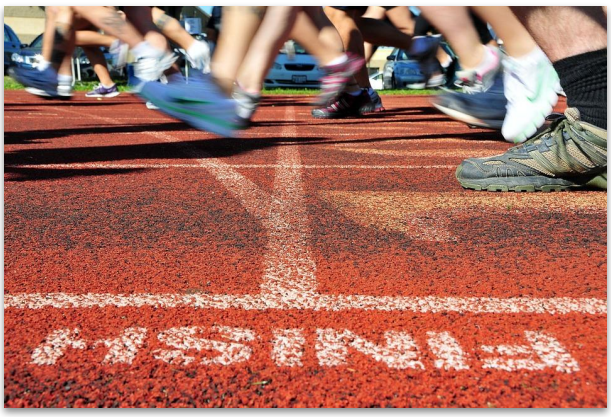
```
ContinueCountdown = {

	countdown = 9,
	continued = false,

	OnStart = function(self)
		Audio.Play(0, "continue", true)
	end,

	OnUpdate = function(self)

		-- Check for continue.
		if self.countdown > 0 and Input.GetKeyDown("space") then
			self.continued = true
		end


		-- Normal countdown
		if self.continued == false then

			if Application.GetFrame() % 60 == 0 then
				self.countdown = self.countdown - 1
			end

			if self.countdown > 0 then
				Text.Draw("continue?", 500, 20, "default", 16, 255, 255, 255,
	255)
				Text.Draw(self.countdown, 500, 100, "default", 50, 255, 255,
	255, 255)
				Image.DrawUI("spot_continue", 100, 100)
			else
				Text.Draw("GAME OVER!", 500, 100, "default", 16, 255, 255,
	255, 255)
				Image.DrawUI("spot_giveup", 100, 100)
			end
		end


		-- Celebrate!
		if self.continued then
			Image.DrawUI("spot_retry", 100, 100)
		end
	end



}
```

20

__/10

100m Dash

A popular contest throughout the world, the 100-meter dash sees athletes sprinting at top speed from a starting line to a finish line. The first to travel 100 meters is declared the victor, while those who violate lane integrity (step out of their lane) are disqualified.



First runner past 100m wins.

Each contestant must stay within a lane.

Objective

Write a Lua component to referee (judge) a basic 100-meter dash race.
Note : If a requirement is ambiguous, it is your choice (different solutions may be valid).
Note : Use the glossary at the front of this exam for a reminder of Lua functions / stuff.

Requirements

☐ Write your Lua component in the box on the following page.
☐ Component must be valid Lua code.
☐ Component makes use of several Lua lifecycle functions from lecture / homeworks.
☐ You may assume multiple actors exist within the scene with the name “runner”
(no actors will be instantiated or destroyed during runtime)
☐ You may assume each runner actor has a Transform component with “.x” and “.y”
(other components in the game will update these values to make the runners move).
☐ The race does not begin (no update logic) until global variable “go” is true (it begins false).
(a different component in the game will set it to true. Until then, it will be false).
☐ At the very end of every frame (after all other components have had a chance to run), check each runner‘s .x and .y position in the order returned by an Actor.FindAll() call.
☐ **Note :** All other components in the codebase use OnUpdate() for their logic.
☐ If any runner has a .x >= 100, declare them the winner (via Debug.Log()) and then Application.Quit() immediately. Assume each runner begins at .x = 0 (the start line).
☐ If any runner has a .y that is more than 0.5 away from their index in the table returned by Actor.FindAll(), declare that runner disqualified (Debug.Log()) and never check them again (**careful**– altering a container while iterating through it can lead to bugs).
☐ Component avoids polluting the global lua state, creating local / table variables only.

Lua and Composition

resources/component_types/DashReferee.lua

```
DashReferee = {

      -- A data structure to track which runners are DQ'd.
      disqualified_runners = {},

      OnStart = function(self)
            -- Collect a reference to all runner actors in the game.
            self.runners = Actor.FindAll("runner")

            -- mark all runners as being "not DQ'd"
            for index, actor in ipairs(self.runners) do
                  self.disqualified_runners [index] = false
            end
      end,

      OnLateUpdate = function(self)
            if go == false then -- "go" is a global here.
                  return -- early out
            end

            -- Check the runners
            for i, actor in ipairs(self.runners) do

                  if self.disqualified_runners[i] == true then
                        continue
                  end

                  local transform = actor:GetComponent("Transform")
                  if math.abs (i - transform.y) >= 0.5 then
                        -- disqualified!
                        Debug.Log("Actor " .. i .. " disqualified")
                        self.disqualified_runners[i] = true
                  else
                        if transform.x >= 100 then
                              -- winner!
                              Debug.Log("Actor " .. i .. " wins!")
                              Application.Quit()
                        end
                  end
            end
      end

}
```

21

___/ 9

A game designer has used your engine to make a game, and sends you the source as a keepsake. You've officially "made it" as an engine developer.

The game's Lua is bug-free and functions as-expected when run. No tricks. Upon inspecting the "resources" folder that represents the game, you find–

resources/images


circle.png


square.png


ko.png

resources/actor_templates/player.template

```
{
  "name": "player",
  "components": {
    "1t": {
      "type":"Transform",
      "x":2
    },
    "2kc": {
      "type":"KeyboardControls"
    },
    "3sr": {
      "type":"SpriteRenderer",
      "image": "circle"
    }
  }
}
```

resources/actor_templates/enemy.template

```
{
  "components": {
    "1t": {
      "type":"Transform",
      "x":9, "y":-2
    },
    "2ai": {
      "type":"EnemyAI"
    },
    "3sr": {
      "type":"SpriteRenderer",
      "image": "square"
    },
  }
}
```

resources/component_types/KeyboardControls.lua

```
KeyboardControls = {
  OnStart = function(self)
    self.t = self.actor:GetComponent("Transform")
  end,

  OnUpdate = function(self)
    if Input.GetKeyDown("up") then
      self.t.y = self.t.y - 1
    end

    if Input.GetKeyDown("down") then
      self.t.y = self.t.y + 1
    end
  end
}
```

resources/component_types/Transform.lua

```
Transform = {
  x = 0,
  y = 0
}
```

Engine Architecture and Lifecycle Functions

resources/component_types/EnemyAI.lua

```
EnemyAI = {
  OnStart = function(self)
    self.t = self.actor:GetComponent("Transform")
  end,

  OnUpdate = function(self)
    if Application.GetFrame() % 2 == 0 then
      self.t.x = self.t.x - 1
    end

    local player = Actor.Find("player")
    local pt = player:GetComponent("Transform")
    if self.t.x == pt.x and self.t.y == pt.y then
      local psr = player:GetComponent("SpriteRenderer")
      psr.image = "ko"

      local pk = player:GetComponent("KeyboardControls")
      pk.enabled = false

      Actor.Destroy(self.actor)
    end
  end
}
```

resources/component_types/SpriteRenderer.lua

```
SpriteRenderer = {

  image = "",

  OnUpdate = function(self)
    local t = self.actor:GetComponent("Transform")
    Image.Draw(self.image, t.x, t.y)
  end

}
```

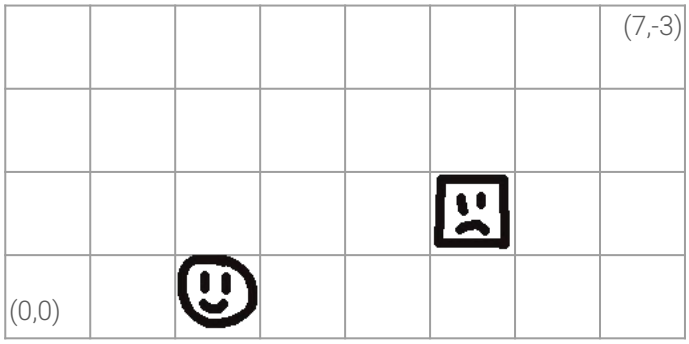
resources/scenes/level1.scene

```
{
  "actors": [
     {
          "name": "player",
          "template": "player"
     },
     {
          "name": "baddie1",
          "template": "enemy",
          "components": {
               "1t": {
                    "x": 6,
                    "y": -1
               }
          }
     },
     {
          "name": "baddie2",
          "template": "enemy",
     },
     {
          "name": "npc"
     }
  ]
}
```

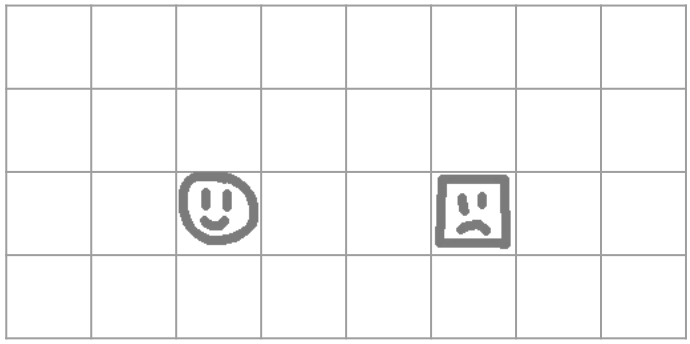
Objective

Draw the first 10 frames of the game. Take note of the input, if any.

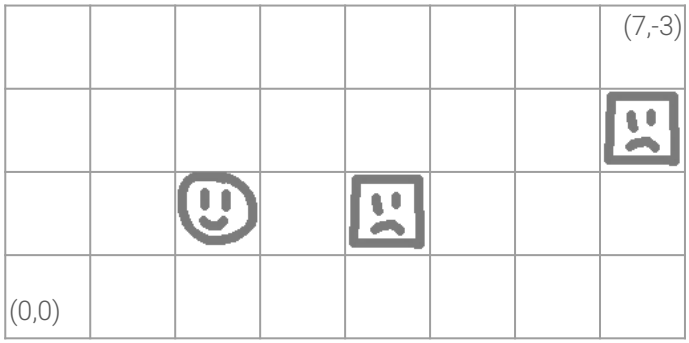
- The first frame is provided for you below.
- The “initial_scene” is “level1.scene”
- Recall that the “render” is the very last thing to occur in a frame (after all logic).
- **Note :** The camera is positioned as indicated by the grid and small position texts.



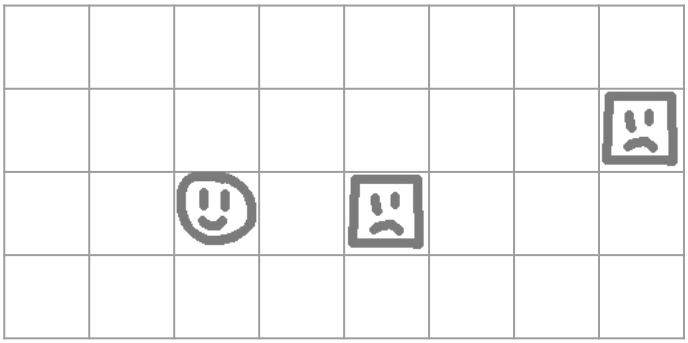
Frame #0 (input held this frame : none)



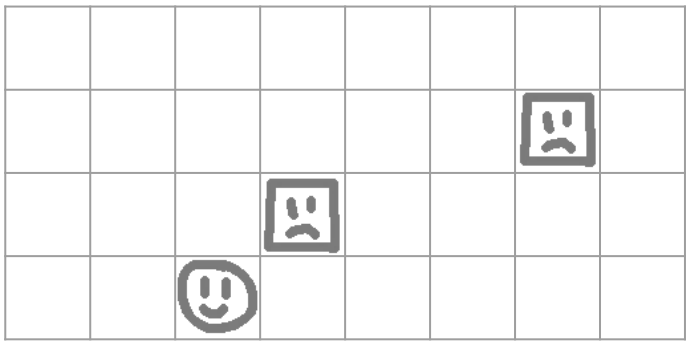
Frame #1 (input held this frame : up)



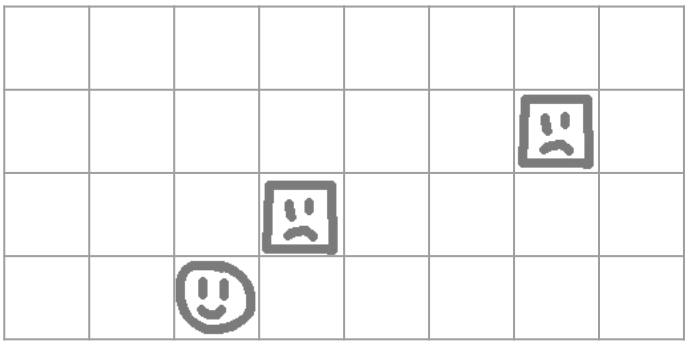
Frame #2 (input held this frame : up)



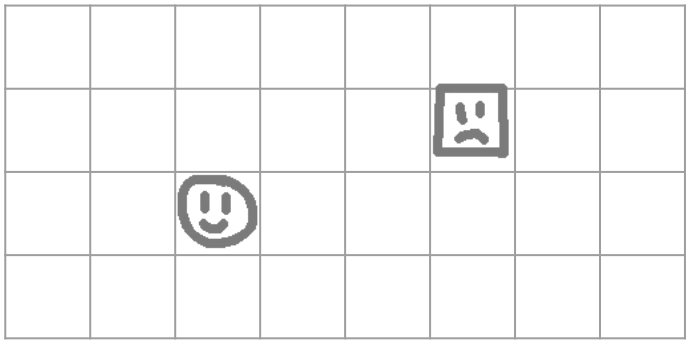
Frame #3 (input held this frame : none)



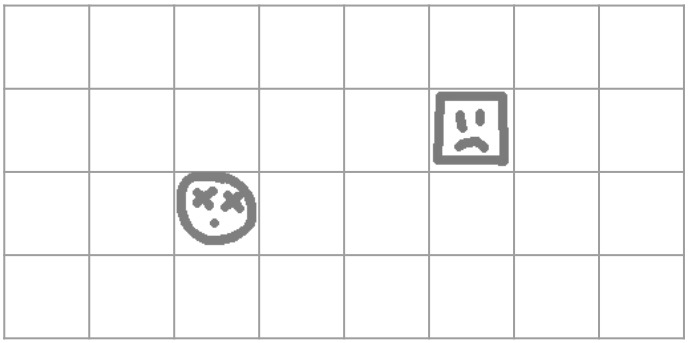
Frame #4 (input held this frame : down)



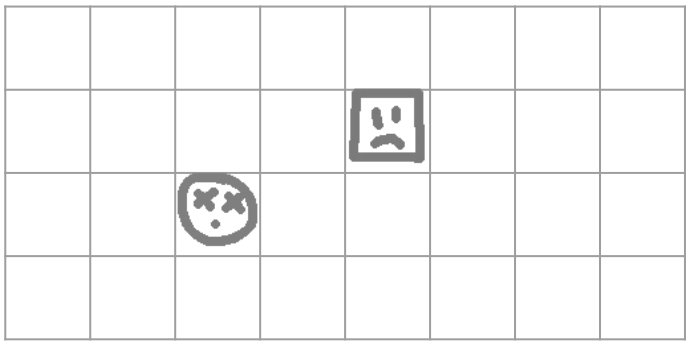
Frame #5 (input held this frame : down)



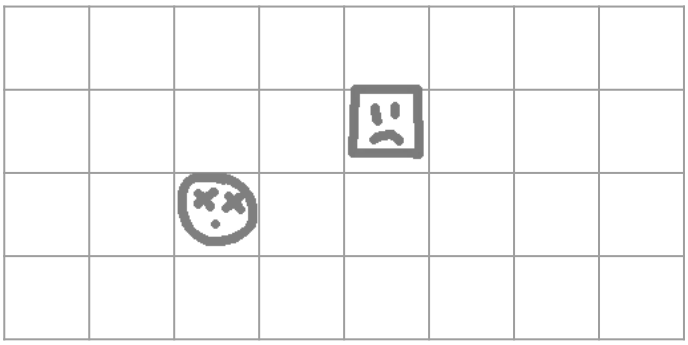
Frame #6 (input held this frame : *up*)



Frame #7 (input held this frame : *none*)



Frame #8 (input held this frame : *up*)



Frame #9 (input held this frame : *none*)

22

__/10

Lifecycle Functions : OnEnteredCameraArea, OnExitedCameraArea

Game designers have begun using your engine to create impressive 2D games.

There is, however, a common request. Game designers want the ability to run code when an actor enters the camera area, and again when an actor leaves the camera's area (ie, when a draw-request associated with an actor enters or leaves the camera's rectangle). You aim to please your game designers, and agree to implement these lifecycle functions such that components like the following will function in a reasonable way--

```
MoveWhenUnseen = {
      -- The monster only follows when its draw call is outside the camera rect.
      seen_by_camera = false,

     -- Called when we go from out of rect to overlapping / inside rect.
      OnEnteredCameraArea = function(self)
            self.seen_by_camera = true
      end,

      -- Called when we go from inside / overlapping rect to totally outside
rect.
      OnExitedCameraArea = function(self)
            self.seen_by_camera = false
      end,

      -- Auto-called every frame we're active / enabled (of course).
      OnUpdate = function(self)
            if self.seen_by_camera == true then
                  return -- early out. Do not move if visible.
            end

            -- AI logic to follow the player around, etc etc.
      end

}
```

Fortunately, your engine code is well-structured to support a new feature like this.

Study the next several pages (a simplified version of our standard course engine), as they contain existing engine files relevant to your task.

Engine Architecture and Lifecycle Functions

game_engine_uniqname/src/Actor.h

```
class Actor
{
public:
	...
	void Update();
	void LateUpdate();

	// A function that checks if visibility lifecycle functions should run
	// and then runs them if so.
	void ConsiderCamLifecycleFunctions(); // TODO : implement this later.




private:
	...

	// Relevant collections of components for this new feature.
	// These are auto-filled for you when components are added to an actor.
	std::map<std::string, luabridge::LuaRef>
components_with_onenteredcameraarea;
	std::map<std::string, luabridge::LuaRef>
components_with_onexitedcameraarea;

	bool visible_last_frame = false;

};
```

game_engine_uniqname/src/Engine.cpp

```
void EngineUpdate() // This gets called automatically once per frame.
{
	for (auto & actor : Scene::GetAllActiveActors())
		actor.Update();




	for (auto & actor : Scene::GetAllActiveActors())
		actor.LateUpdate();


	for (auto & actor : Scene::GetAllActiveActors())
		actor.ConsiderCamLifecycleFunctions();

	RenderAllDrawRequests(); // performs all rendering for the frame.
}
```

game_engine_uniqname/src/DrawUIRequest.h

```
// A data structure that represents each Image.DrawUI request.
// Use GetRequestForActor() to inspect the current frame's draw requests.
class DrawUIRequest
{
public:
      Actor* requesting_actor // The actor that is being drawn.
      int x; // The x pos (in screen-space pixels) of top-left corner of the
draw.
      int y; // The y pos (in screen-space pixels) of top-left corner of the
draw.
      int w; // The width (in screen-space pixels) of the draw.
      int h; // The height (in screen-space pixels) of the draw.

      // Call this function to get the draw request in the current frame for
      // a particular actor. Returns nullptr if no request this frame.
      static DrawUIRequest* GetRequestForActor(Actor* actor);
};
```

game_engine_uniqname/src/EngineUtil.h

```
class EngineUtil
{
public:
      static int GetCamWidth(); // Returns camera / window width in pixels.
      Static int GetCamHeight(); // Returns camera / window height in pixels.
};
```

Objective

Add a very small amount of state to **Actor.h** and code to **Engine.cpp** on the previous page. Then fill out **Actor.cpp's** ConsiderCamLifecycleFunctions() on the next page, so to bring the OnEnteredCameraArea and OnExitedCameraArea Lua lifecycle functions to life.

- **Note :** All draw requests have an Actor* field now– the actor that is being drawn.
- **Tip :** View the glossary at the front of the exam for possibly-useful C++.
- We only care about screen-space / UI requests in this problem (no scene-space).
  - ie, you need not worry about any camera movement, zooming, etc.
- This question deals in c++-like pseudo code. Your syntax will be accepted so long as your intentions and logic are very clear, and it resembles valid c++.
- When an actor comes into existence, consider its initial state "invisible" by default.
  - ie, first frame an actor does a draw request, we call OnEnteredCameraArea().
- These new lifecycle functions should be called after OnLateUpdate() in the frame.
- You may assume any given actor will make **one draw request MAX** per-frame.
- Recall that in our engines, (0, 0) = top left of window. (width, height) = bottom right.

Engine Architecture and Lifecycle Functions

game_engine_uniqname/src/Actor.cpp

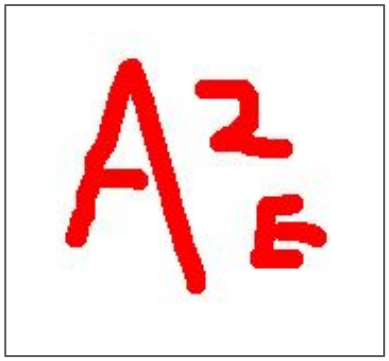
```
void Actor::ConsiderCamLifecycleFunctions()
{
	bool visible_this_frame = false;

	DrawUIRequest* req = DrawUIRequest::GetRequestForActor(this);

	if (req != nullptr)
	{
		// Oh my god, that's AABB's music!!
		int cam_w = EngineUtil::GetCamWidth();
		int cam_h = EngineUtil::GetCamHeight();

		if (req->x > cam_w || req->y > cam_h || req->x + req->w < 0 || req->y +
	req->h < 0)
			visible_this_frame = false;
		else
			visible_this_frame = true;
	}

	/* If our state differed from last frame, we need to call lifecycle functions
*/
	if (visible_this_frame == true && visible_last_frame == false)
	{
		// We became visible this frame.
		for (auto & component : components_with_onenteredcameraarea)
		{
			component.second["OnEnteredCameraArea"](component.second);
		}

	}
	else if (visible_this_frame == false && visible_last_frame == true)
	{
		// We became invisible this frame.
		for (auto & component : components_with_onexitedcameraarea)
		{
			component.second["OnExitedCameraArea"](component.second);
		}
	}

	// Prep for next frame.
	visible_last_frame = visible_this_frame;
}
```

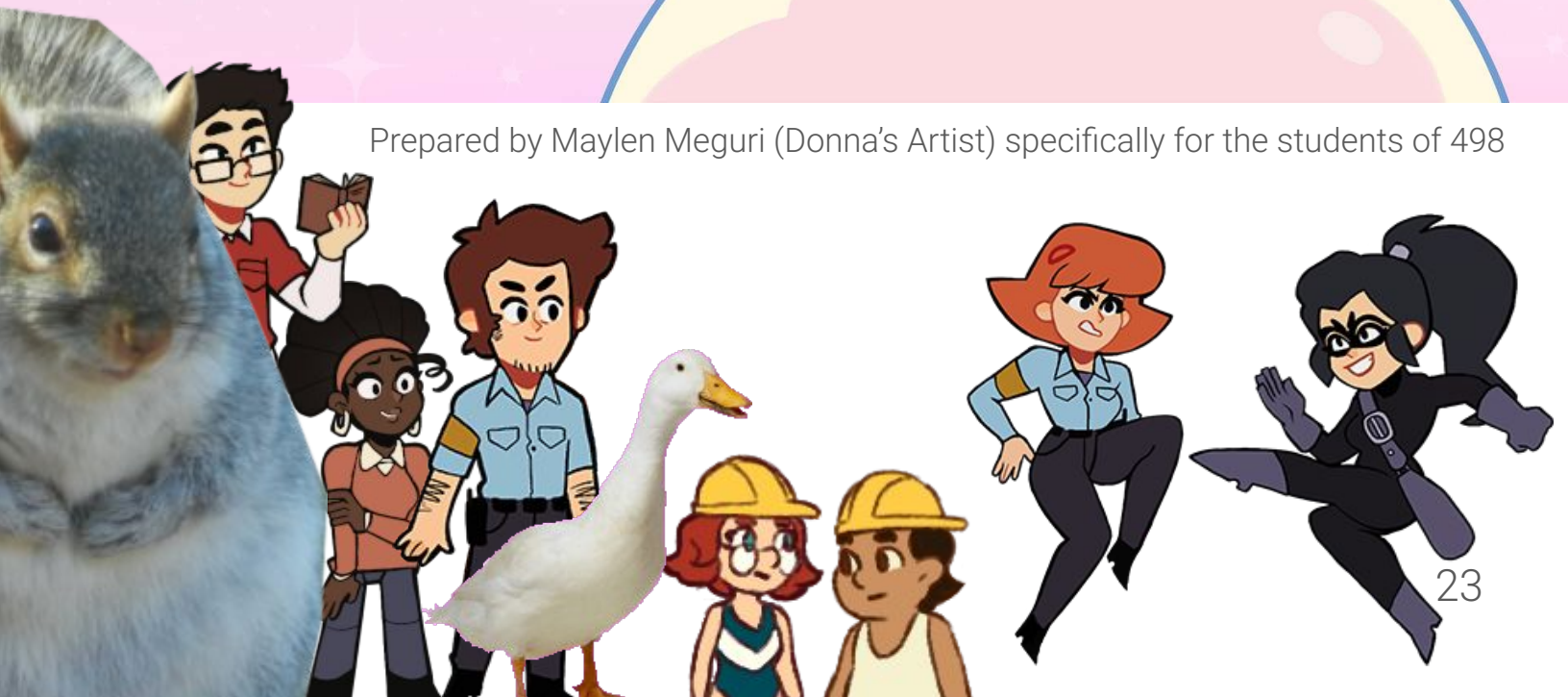
23

__/1

Name your new engine and draw its logo ->
(you need not keep the name “A2 Engine”)

A2 Engine





Prepared by Maylen Meguri (Donna’s Artist) specifically for the students of 498