Winter 2026 Mid-Semester Exam - Question Packet

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

Glossary

Helpful Engine Lua API Functions
Image.DrawUI(string image_name, int x, int y) – Draw an image at (x,y) in screen space.
Image.Draw(image_name, int x, int y) – Draws image in scene space (x,y) (**note :** no floats)
Image.DrawEx(image_name, int x, int y, rotation, scalex, scaly, pivot_x, pivot_y, r, g, b, sort)

Text.Draw(content_str, x, y) -- Draw the text “content_str” at position x, y

Audio.Play(int channel, string clip_name, bool does_loop)

Actor.Instantiate(template_name) -- Instantiates an actor of a template and returns a ref.
Actor.Find(string name) – Returns a single actor reference by name.
Actor.FindAll(string name) – Returns all actors of a given name.
actor_ref:GetComponent(component_type_name) – Get reference to a component by type.

Application.GetFrame() --Returns the current frame of the game.
Application.Quit() -- Terminates the game process.

Input.GetKey(key) -- Returns true if the key is currently down on this frame.
Input.GetKeyDown(key) – Returns true if the key was pressed down this frame.

Event.Publish(event_type, event_object) -- “event_type” is a string
Event.Subscribe(event_type, component_ref, function_ref)
Event.Unsubscribe(event_type, component_ref, function_ref)

nil --Lua’s typical value to represent “null” or “nothing” or “invalid”

table.supersort(some_table, function(a, b) -- returns a new table (array) with items sorted.
    return a.some_property < b.some_property -- This results in ascending order.
end)

Helpful Common Lua Functions
table.insert (my_table, thing_to_add) – Adds a thing to the very end of a table.
table.remove(my_table, index_to_remove) -- Removes the item at “index_to_remove”
#table_name -- Get the number of items in a table.

Kim Possible / KP

```
-- example of an event-handling member function inside a component.
-- Register to handle event with Event.Subscribe("event_type", self, self.EventHandler)
EventHandler = function(self, event_object)
        -- Do some logic!
end
```

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.

```
if <condition> then
-- Do Stuff
elseif <confiditon> then
-- Do Stuff
else
-- Do Stuff
end

local new_string = "hello " .. "world" --The string concatenation operator
```

Helpful C++ / Pseudocode Techniques

```
my_function_luaref (component_luaref); // call a lua function from c++ (with self-reference).

vec.erase(vec.begin() + index); // Removing an element from an std::vector by index.

vec.push_back(item); // Add an item to an std::vector

for (LuaRef & lua_thing : lua_refs) { } // for-each iteration.
for(int i = 0; i < vec.size(); i++) { } // for-index iteration.
break; // break out of a loop.

int result = static_cast<int>(2.1f / 1.0f); // Get the integer portion of a division operation.

try { /* some logic that might cause a problem */ }
catch (const LuaException& e) { /* some logic */ }
```

16

__/10

A Sitch in the UI

When it comes to important gameplay information (health, lives, cooldowns, experience, ammo, etc), players need timely information to make good decisions.

One classic method is UI (sometimes known as HUD - “Heads Up Display”). Beyond serving information, designers may inject character and personality via UI.

Kim Possible for the Gameboy Advance (2002), a classic-style spy-themed platformer created by one of Austin’s friends, Kevin James.

bar.png (each of 5 red vertical bars)

UI indicates health (bar) and lives (number). Note the face change as a function of health. The expression grows annoyed with damage.

intro.png

A pre-stage briefing image covers the entire UI in the moments before gameplay begins, establishing narrative and context.

gameover.png

An ending image covers the entire UI after lives are exhausted, concluding the experience.

Lua and Composition

Objective
Write a “KPManager” Lua component to handle gameplay and UI-related state.

Note : If a requirement is ambiguous or absent = your choice (different ideas may be valid).
Note : Use the glossary at the front of this exam for a reminder of Lua functions / technique.

Requirements
☐ The Lua component is written in the corresponding box on your answer document.
☐ All lua code is valid / syntactically correct. Exceptions are not generated.
☐ Component makes use of Lua lifecycle function(s) from lecture / homeworks.
☐ Component creates only local / member variables and avoids polluting the global lua state.

Intro
☐ When the game begins play audio “intro.wav” on channel 5 with no looping.
☐ For approximately the first 180 frames of the game, render “intro.png” at 0,0 in the UI.

Respawning & Conclusion
☐ Kim begins the game with 3 lives. Track this.
☐ At the very end of every frame, check if an actor named “kp” exists.
    ☐ If the actor does not exist, consider how many lives Kim has remaining.
        ☐ If lives > 0
            ☐ Instantiate the “kp” template.
            ☐ Move the new kp actor to position 3,4 via Transform component (.x & .y)
            ☐ Reduce Kim’s remaining lives by one.
        ☐ If lives <= 0,
            ☐ Render “gameover.png” at 0,0 in the UI every frame.
            ☐ Play “gameover.wav” audio once on channel 5 with no looping.
            ☐ Approximately 180 frames later, conclude the game with Application.Quit()

Health
- The “kp” actor, if it exists, has a “status” component with an “hp” variable that goes from 4 down to 0.
    ☐ Draw one “bar.png” in the UI for each point of hp above 0. Draw at (50+50*i ,50) (“i” is the zeroed index of the health block. When hp is 0, no blocks are drawn)
- Kim’s facial expression is captured in images **4.png**, **3.png**, **2.png**, **1.png**, and **0.png**
    ☐ Draw the facial expression corresponding to Kim’s current hp in the UI at 0,0
    ☐ Draw the text “xL” where L is the number of lives remaining. Do so at 75,75

17

__/10

High Score Hero

The golden era of the arcade (1990s) was also the golden era of local, in-person competition. Players from communities near and far would gather to learn, improve, and one-up their friends on the leaderboards of their favorite games.

By now, you know a thing or two about tables– let's explore "high score" tables.

The leaderboard of Capcom's *Street Fighter 2*. This game, credited for launching the golden era of arcades, features simple rankings with only score alongside player initials.

Virtua Fighter 5's high scores screen. Players see a ranking list with player initials, their character, and more. (image GameUiDatabase.com)

The stylish leaderboard of *Catherine* (2011) is an online, global leaderboard. Players from around the world submit their scores, and it updates accordingly. (image GameUiDatabase.com)

Lua and Composition

Objective
Write a “GlobalHighScoreManager” component to handle a regularly-updating leaderboard.

Note : If a requirement is ambiguous or absent = your choice (different ideas may be valid).
Note : Use the glossary at the front of this exam for a reminder of Lua functions / technique.

Requirements
☐ The Lua component is written in the corresponding box on your answer document.
☐ All lua code is valid / syntactically correct. Exceptions are not generated.
☐ Component makes use of Lua lifecycle function(s) from lecture / homeworks.
☐ Component creates only local / member variables and avoids polluting the global lua state.

Receiving Data
- Events of type “global_score_received” are sometimes published from elsewhere.
- These events always provide an “event_object” with a `.score` number, a `.character` string, and a `.player_name` string.

☐ Track all “global_score_received” event objects as they arrive. No scores will be negative.
☐ If a single player_name has appeared in multiple score objects, only the score object with the highest score is tracked (ie, one score object is tracked per player).

Notification of New Global High Score

☐ If a “global_score_received” event object arrives, and its score is higher than the best score ever yet seen, publish a new event of type “global_new_highscore” with said event object.

Organizing and Displaying the Data

☐ Before rendering, the events are sorted in order of score from highest to lowest.
– Tip : Use table.supersort(), described in the glossary. This returns a brand new table with everything sorted and ready for iteration with ipairs()

☐ Every frame, render all event objects in order.
- For each event object (`i` is the table index of the event object)...
  - ☐ Draw the rank number in the UI at (50, 50 + `i` * 50)
  - ☐ Draw the score number in the UI at (100, 50 + `i` * 50)
  - ☐ Draw the player_name string in the UI at (200, 50 + `i` * 50)
  - ☐ Draw an image in the UI at (300, 50 + `i` * 50) representing the character.
    - The character’s image filename is the `character` string.

18

__ / 9

Using the game engine you developed this semester, you decide to make a simple action game (a fangame) based on the 2000s-era *Kim Possible* IP.

After creating some assets and writing some code, your resources/ looks like–

resources/images

IMPORTANT
You do not need to draw the above sprites perfectly, agent. Draw stick figures instead, but make sure it is clear which one you intended to draw.

Tip : Use the hand / arm / limb poses for grading clarity.

resources/actor_templates/player.template

```
{
  "name": "kim",
  "components": {
    "1t": {
      "type":"Transform",
      "x":1
    },
    "2kc": {
      "type":"KeyboardControls"
    },
    "3g": {
      "type": "PlayerGravity"
    },
    "4sr": {
      "type":"SpriteRenderer",
      "image": "idle"
    }
  }
}
```

resources/actor_templates/enemy.template

```
{
  "components": {
    "1t": {
      "type":"Transform",
      "x":7
    },
    "2spawn" {
       "type":"TemplateSpawner",
       "template_name":"bullet",
       "frame_delay": 4,
       "spawn_x": 6
    },
    "3sr": {
      "type":"SpriteRenderer",
      "image": "enemy"
    },
  }
}
```

Engine Architecture and Lifecycle Functions

resources/actor_templates/bullet.template

```
{
  "components": {
    "1t": {
      "type":"Transform"
    },
    "2v": {
      "type": "ConstantVelocity",
      "x_vel": -1
    },
    "3ko": {
      "type":"KoOnTouch"
    },
    "4sr": {
      "type":"SpriteRenderer",
      "image": "bullet"
    }
  }
}
```

resources/component_types/Transform.lua

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

resources/component_types/KeyboardControls.lua

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

  OnUpdate = function(self)
    if game_state == "gameover" then
      return
    end

    if self.t.y >= 0 then
      if Input.GetKeyDown("space") then self.t.y = self.t.y - 3 end
    end

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

resources/component_types/SpriteRenderer.lua

```
SpriteRenderer = {

  image = "",
  sort = 0,

  OnLateUpdate = function(self)
    local t = self.actor:GetComponent("Transform")

    -- Tip : The function call below draws the image normally…
    -- BUT the pivot point of the sprite is the very bottom center of the sprite.
    Image.DrawEx(self.image, t.x, t.y, 0, 1, 1, 0.5, 1.0, 255, 255, 255, 255, self.sort)
  end
}
```

Engine Architecture and Lifecycle Functions

resources/component_types/PlayerGravity.lua

```
PlayerGravity = {

  OnUpdate = function(self)
    local t = self.actor:GetComponent("Transform")
    local sr = self.actor:GetComponent("SpriteRenderer")

    -- Gravity
    if t.y < 0 then
      t.y = t.y + 1
    end

    -- Animate
    if game_state == "gameplay" then

      if t.y < 0 then sr.image = "jump" end
      if t.y >= 0 then sr.image = "idle" end

    end
  end
}
```

resources/component_types/TemplateSpawner.lua

```
TemplateSpawner = {

  template_name = "???",
  frame_delay = 0,
  spawn_x = 0,
  spawn_y = 0,

  OnUpdate = function(self)

    if game_state == "gameover" then
      return
    end

    -- Spawn the template.
    -- Reminder : The new actor's components don't run until next frame.
    if Application.GetFrame() % self.frame_delay == 0 then --Modulo operation
      local new_actor = Actor.Instantiate(self.template_name)
      local new_actor_t = new_actor:GetComponent("Transform")
      new_actor_t.x = self.spawn_x
      new_actor_t.y = self.spawn_y
    end

  end
}
```

Engine Architecture and Lifecycle Functions

resources/component_types/KoOnTouch.lua

```
KoOnTouch = {

  OnUpdate = function(self)
    local t = self.actor:GetComponent("Transform")
    local player = Actor.Find("kim")
    local player_t = player:GetComponent("Transform")

    if t.x == player_t.x and t.y == player_t.y then
      game_state = "gameover"
      local player_sr = player:GetComponent("SpriteRenderer")
      player_sr.image = "ko"
      player_t.y = player_t.y - 2
      Actor.Destroy(self.actor)
    end
  end
}
```

resources/component_types/ConstantVelocity.lua

```
ConstantVelocity = {

  x_vel = 1,
  y_vel = 0,

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

  OnUpdate = function(self)

    self.t.x = self.t.x + self.x_vel
    self.t.y = self.t.y + self.y_vel

  end
}
```

resources/scenes/level1.scene

```
{
  "actors": [
      {
            "name": "kim",
            "template": "player",
      },
      {
            "name": "shego",
            "template": "enemy"
      }
      ]
}
```

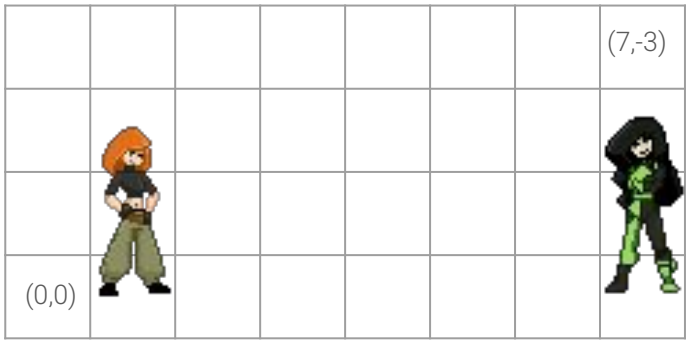
Careful Agent!
These images span multiple cells! Notice **where the bottom-center of the sprite is (the pivot point)!**

Objective

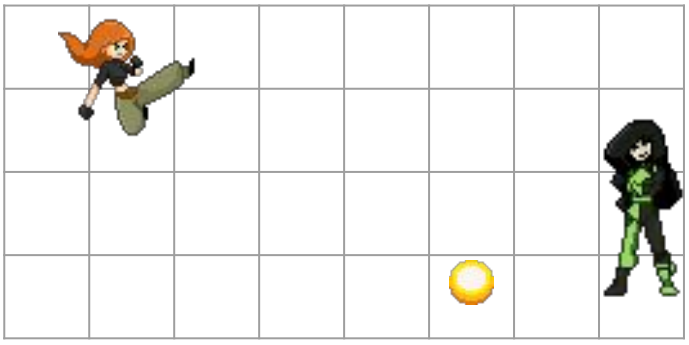
Draw the first 10 frames of the game. Take note of the input before the frame, if any.

Notes--

- The first two frames are provided for you below, and are accurate.
- Stick figure drawings are acceptable. Use limb positions to make clear your choice.
- "Pivot Point" is particularly important in this problem. All images use bottom-center.
- The world is 8x4 in dimension. Coordinates are all in scene space (even UI text).
- The "initial_scene" is "level1.scene". Bottom-left cell is 0,0 while top-right is 7,-3
- The camera defaults to position {3.5, -1.5} as seen below. It will never move.
- Recall that these "renders" are the last thing to occur in a frame (after all logic).
- Actors occlude each other if they stand in the same cell. No transparency.



Frame #0 (input coming early frame : **none**)



Frame #1 (input coming early frame : **space**)

19

__/10

“Lifecycle” Function : Audio.RegisterBeat () + Audio.UnregisterBeat ()

Due to your phenomenal work in EECS 404, you have been hired by Nintendo. Congratulations!

Nintendo products are acclaimed for their iconic music, produced by a long-time, in-house audio team and a culture of audio-prioritization.

In order to better support the creation of rhythm-focused games such as *Rhythm Heaven* (2008) and *Donkey Konga* (2003), as well as rhythm-related features in other games such as *New Super Mario Bros* (2006), Nintendo’s game designers request the addition of **two beat-based lifecycle functions.**

Your help will be needed on the C++ side to bring components like this to life :

```
ScaleLargeToTheBeat= {
      channel = 5,

      OnStart = function(self)
            self.transform = self.actor:GetComponent("Transform")

            -- We register a a function to be called on the beat.
            Audio.RegisterBeat(self.channel, self.OnBeat)
      end,

      OnBeat = function(self)
            -- The beat has arrived! Change our scale to be larger.
            self.transform.x_scale = 2.0
            self.transform.y_scale = 2.0
      end,

      OnUpdate = function(self)
            -- If our scale is larger than normal, shrink back to normal slowly.
            if self.transform.x_scale > 1.0 then
                  self.transform.x_scale = self.transform.x_scale - 0.01
                  self.transform.y_scale = self.transform.y_scale - 0.01
            end
      end,

      OnDestroy = function(self)
            -- This component is finished, so let’s un-register.
            Audio.UnregisterBeat(self.channel, self.OnBeat)
      end
}
```

Engine Architecture and Lifecycle Functions

Your objective
Bring these "lifecycle" functions to life and call all registered functions when their channel reaches a new beat.

Look over the files and functions for this problem in the answer packet. Then…

☐ Decide on two data structures to add to AudioDB class.
- ☐ A data structure to track your function registrations per-channel. (necessary to call the right functions)
- ☐ A data structure to track the "previous beat" for a channel. (necessary to know when a new beat was reached)

☐ Write the AudioDB::RegisterBeat() function to track registrations
☐ Write the AudioDB::UnregisterBeat() function to remove registrations
☐ Write the AudioDB::OnUpdate() function.
(This function runs every frame automatically)
- ☐ This function must check all active channels.
- ☐ For a given active channel, decide if a new beat has been reached by looking up some information on the channel and the audio file it is playing, and considering the timestamp of the previous beat (your data structure).
  - **Tip :** Integer division may be used to find a channel's current beat count. (check glossary)
    - ☐ Do something to prevent lua-code errors from crashing the engine.
- ☐ If no new beat is reached for a channel, nothing happens for that channel.
- ☐ If new beat has been reached, call all lua functions registered for that channel.

Notes
- You may handle the very first beat of a channel however you like.
- Assume LuaRef is copyable / easy to pass around without side effects.
- Assume LuaRef has equality (==) defined, but that it is **NOT hashable.**
- Do not worry about the state of the actor (active) or components (enabled).
- Do not worry about an active channel restarting or looping– they won't do this.
- Do not worry about static data structures needing definition (inline).
- Do not worry about issues caused by removal of registrations mid-frame.
- Assume no lua function will be registered on the same channel more than once.
- A LuaRef representing a function's component may be obtained via the `GetComponentOfFunction()` function.

Hints
- Check the glossary's C++ / LuaBridge section.
- Review all of the functions and data structures the question provides to you.