COMP2611: Computer Organization, Spring 2016

Programming Project: Survival of the Quickest

(Submission deadline May 9 at 5pm via CASS)

1. Introduction

In this project, you will implement (in part) a student survival game, called “Survival of the Quickest” using MIPS assembly language.

Figure 1 shows a snapshot of the game. The game displays a maze, where various score points appear at random and disappear after a while. The player moves a student character around to collect as many score points as possible. Multiple teacher characters wander around the maze. When the student bumps into a teacher, he/she needs to answer a quiz question. The student loses points if the answer is wrong, otherwise she/he become shielded from the teachers temporarily with a ‘cloak of invisibility’, allowing her/him to cross the teachers without having to answer questions.

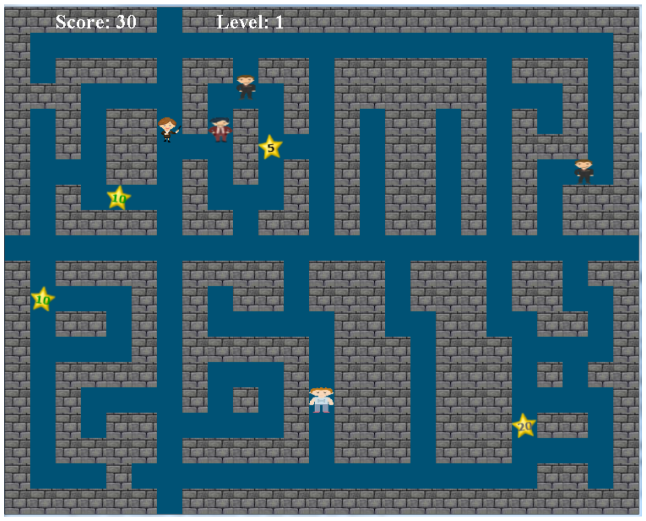


Figure 1. Survival of the Quickest.

2. Coordinate System

The game screen size is of 750 x 600 pixels with the origin being the top left corner as illustrated in Figure 2. Pixels in the game screen are arranged in a two-dimensional array of integers, each element being one pixel (or one colored dot). The top-left corner pixel being (0, 0), the bottom-right corner pixel is consequently (749, 599).

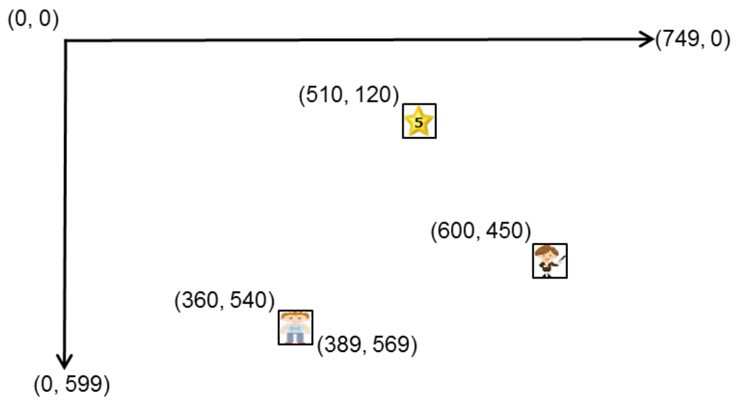


Figure 2. The coordinate system.

Each object in the game is represented by a rectangular image, and its location is referenced by its top-left corner coordinates. For example, the student character in Figure 2 is at (360, 540) (the image size here being 30x30 pixels).

The maze is of the same size as the game screen, with 20-rows by 25-columns grid of cells. Each cell is 30 x 30 pixels. A cell is either a wall ▦ or an open path ■. The maze is encoded as a two-dimensional bitmap array of 1s and 0s representing walls and paths respectively as shown in Figure 3.

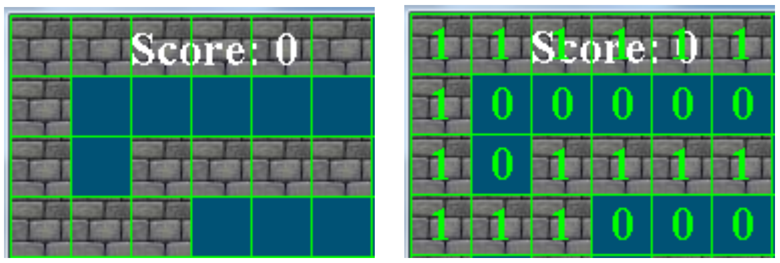


Figure 3. Illustration of the maze's grid (green square) and bitmap.

3. Game Objects

There are three types of game objects: student, teacher and score point. Every object has attributes as listed below.

- **Current location:** a (x, y) coordinate indicating the current location of the object.
- **Moving speed:** an integer variable indicating the moving speed of the game object. **Score Value (SV):** an integer variable indicating the score value of a score point object.

Object	**Width**	**Height**	**Speed (pixel)**	**SV**	**Initial location**
Student (or the like)	30	30	3	-	(360, 540)
Teacher (, , or the like)	30	30	2	-	Random path cell (far from Student). Teachers do not overlap.
Score point (or the like)	30	30	-	Random	Random path cell (far from Student). Score points do not overlap.

Table 1: Images, sizes and attributes of the game objects

4. Game Details

4.1 Game initialization

Player chooses the number of teacher objects and score point objects (value 1 to 5) at the beginning of the game. Then, the teacher objects, score point objects and one student object are created with images retrieved from an image repository. The game screen is then created and displayed. The game has two levels: Level 1 and Level 2.

Other initial settings at the beginning of each level include:

- The location of the student object is set to (360, 540);
- Each teacher object is put on a random path cell of the maze far from the student object, and no two teacher objects overlap;
- Each score point object is put on a random path cell of the maze far from the student object, and no two score point objects overlap;

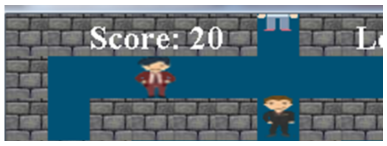
- The initial game score is 30;

Background music is played throughout the game. There are various sound effects for different game events (e.g. the player obtaining score points).

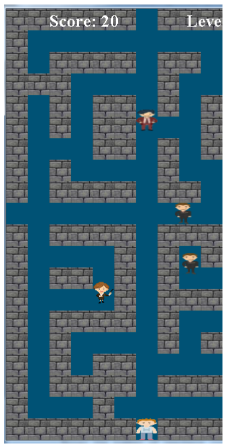
4.2 Object Movements

Student object moves vertically or horizontally when the player press “a”, ”s”, ”d”, “w”. Each keystroke moves the student object by 3 pixels for 10 game iterations. The is no movement when the student bumps a wall.

The student can cross the border if there is a path (i.e., paths are toroids like the K-Map). This is shown in Figure 4 (a) where the student crossed the border at the top to re-appear at the opposite border (at the bottom) as shown in Figure 4 (b).



(a)



(b)

Figure 4. (a) Student partially crosses the maze’s upper border. (b) Student is relocated to the maze’s bottom border.

During the game, the teacher objects patrol the maze in preset routes (by the game AI engine). Teachers in Level 2 are smarter (or vampiric). If the teacher is far away from the student, she/he teleports (with 1% probability) to a location nearby or even the same as that of the student.

4.3 Score Points

The score point objects appear (i.e., visible) and disappear (i.e., invisible) at random path cells during the game. Each score point object carries a score value (SV), which is also shown to the player.

4.4 Collision Between Game Objects

Two game objects collide with each other when their images (rectangles) intersect. Figure 5 shows possible collision scenarios:

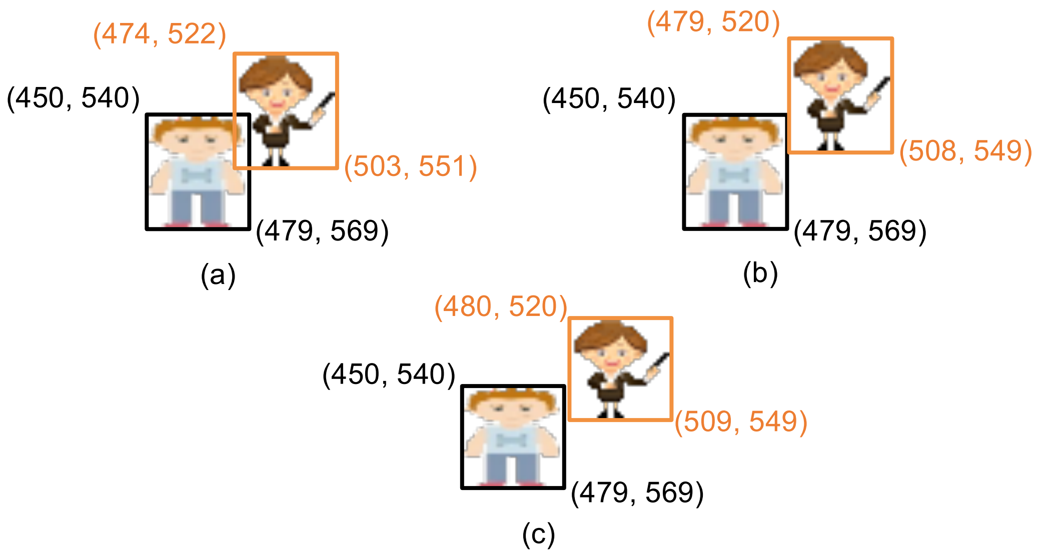


Figure 5. (a) Collision. (b) Collision. (c) No collision.

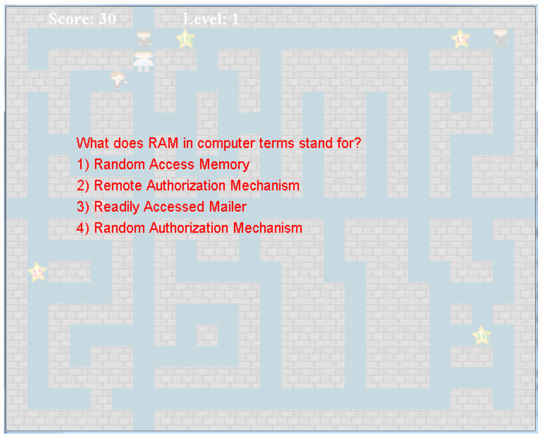
- Collision of the student and a visible score point object: Score! SV is added to game score. Make score point object invisible.
- Collision of the student and a teacher object: Quiz mode is triggered (described in Section 4.5).
- Collision of the student and multiple teacher objects at the same time: Quiz mode is triggered only once.
- Collision of the student, a teacher object and a score point object at the same time: take the score first, then enter into the quiz mode if the game score is still less than 100.

4.5 Shield mode and Quiz mode

Collision of student and teacher object triggers "quiz mode". During quiz mode:

- A MC quiz question is displayed (see an example in **Error! Reference source not found.**);
- All the teacher objects stop moving;

- The countdown of the remaining time of all the score point objects in visible or hidden status pauses;
- Any inputs of the player other than those for answering the quiz are not processed.



What does RAM in computer terms stand for?
1) Random Access Memory
2) Remote Authorization Mechanism
3) Readily Accessed Mailer
4) Random Authorization Mechanism

Figure 6. The Quiz mode.

- Answering a quiz question correctly puts student in "shield mode" (represented as or the like). Shield mode lasts for a period of time (500 student movements) and then everything goes back to normal.
- When the student is protected by a 'shield', no teacher can catch her/him. The collision does not trigger the quiz mode.
  Answering quiz question wrongly will deduct game score of the player, and it will also cause the student to teleport to a random open-path of the maze without overlapping with any score point object or teacher object there.

4.6 Winning and Losing the game

During the game play, if the game score reaches 0 or less, the player loses the game. If the game score reaches 100 or more, the player will pass Level 1 and promote to Level 2. The player wins the game if both levels are passed.

5. Game Skeleton

A skeleton file `comp2611_Spring2016_project_skeleton.s` is provided to you. After the game initialization, the game will proceed in a loop, whose iteration is quoted as game iteration. Each iteration follows the steps listed below:

1. Get the current time: `jal get_time`

2. Get the keyboard input: `jal get_keyboard_input`

   Note: Holding down the key generates a sequence of keystrokes, which moves student object continuously. For MAC users, if it does not work, enable it by entering the following command in the Terminal application: "`defaults write -g ApplePressAndHoldEnabled -bool false`". Replacing `false` by `true` in the command will produce the opposite effect.

3. If the game is in the Quiz mode, go to step 13.

4. Check collisions with score point objects: `jal check_scorepoint_collisions`. For each visible score point object, check whether the student object has collided with it. If a collision has happened, then update the game score, hide that score point object and skip further collision checking with any other score point objects.

5. Check whether the game level reached the winning condition, and perform the corresponding action: `jal check_level_end`. If the player wins the game, terminate the game. If the game has promoted to Level 2, go to step 1, otherwise go to step 6.

6. If the game is in the Shield mode, go to step 8.

7. Check collisions with teachers: `jal check_teacher_collisions`. Check collision among student-teacher for every teacher object. If there is a collision, trigger the Quiz mode and go to step 16 afterward, skip further collision checking with any other teacher objects.

8. Move all the teacher objects: use the `syscall 213`.

9. Update the visible or hidden status of all the score point objects: `jal update_scorepoint_status`.

10. Process the current keyboard input for changing the game status: `jal process_status_input`. If the input is valid, perform the corresponding action.

Valid input	Action
m	Toggle the playing of the background music.
0	If the game is not in the Shield mode, trigger the Shield mode. This is a cheat code built for you (in case you can't answer quiz questions) to enforce the shield mode for easy debugging.

11. Continue any previous student movement or process the latest movement input: `jal process_move_input`. First, save the current keyboard input as the latest movement input if it is a valid input as shown in the table below. Then, if a 10-iteration student movement started in a past iteration has not finished yet, continue the movement, otherwise perform the corresponding action of the saved latest movement input (if any) and then remove this saved input.

Valid input	Action
w	Start the upward movement of the student object (moving it upward by its speed for 10 game iterations starting from the current iteration, and if the game is in the Shield mode, also update the mode's status, using `jal update_shield_status`, in each of those iterations).
a	Start the leftward movement of the student object.
s	Start the downward movement of the student object.
d	Start the rightward movement of the student object.

12. Go to step 16.

13. Process the current keyboard input for the Quiz mode: `jal process_quiz_input`. If the input is valid, perform the corresponding action.

Valid input	**Action**
1	Answer the quiz by its answer choice 1 (end the Quiz mode, check the correctness of the answer and update the game score, trigger the Shield mode and/or perform the student teleportation based on that correctness).
2	Answer the quiz by its answer choice 2.
3	Answer the quiz by its answer choice 3.
4	Answer the quiz by its answer choice 4.

14. Check whether the game reaches the losing condition, and perform the corresponding action: `jal check_level_end`. If game is over, terminate the game.
15. If the game is in the Quiz mode, then update the status of the mode: `jal update_quiz_status`.
16. Refresh the game screen to reflect any screen changes: use the `syscall 201`.
17. Take a nap: `jal have_a_nap`. The interval between two consecutive game iterations of this loop is usually about 30 milliseconds.
18. Go to step 1.

6. Tasks to Complete

Read the skeleton code and understand how it works. Complete the following MIPS procedures. You should not modify the skeleton code but only add your code to those procedures.

Note: You don't need to understand every single detail of the skeleton code. You can discuss with your friends if you have difficulty in understanding it. But every single line of your code should be your own work (not something copied from your friends).

Procedure	**Inputs**	**Outputs**	**Description**
`move_student_left`	$a0 = 1 or 0	$v0 = 1 if a movement has been made, or else 0	Move the student object leftward by its speed for one game iteration. If $a0 is not 0, do not move the object if it will overlap with a wall cell in the movement.
`move_student_right`	$a0 = 1 or 0	$v0 = 1 if a movement has been made, or else 0	Move the student object rightward by its speed for one game iteration. If $a0 is not 0, do not move the object if it will overlap with a wall cell in the movement.
`check_intersection`	RectA: ((x1,y1), (x2, y2)) RectB: ((x3,y3), (x4, y4))	$v0: 1 for true (intersection happened); 0 for false	Check whether the given two rectangles intersect. (x1, y1) and (x2, y2) are the coordinates of top-left and bottom-right corners of rectangle RectA. (x3, y3) and (x4, y4) are the coordinates of top-left and bottom-right corners of rectangle RectB. The eight coordinates are passed via the stack.
`check_teacher_collisions`		$v0 = 1 if a collision has been found, or else 0	Check whether the student object has collided with a teacher object. This procedure pushes the coordinates of two game objects into the stack, and then calls the procedure check_intersection to detect whether they intersect each other.
`check_scorepoint_collisions`			Check whether the student object has collided with a

			score point object. If a collision has been found, then hide the score point object, increase the game score by its SV and skip further collision checking with any other score point objects. This procedure pushes the coordinates of two game objects into the stack, and then calls the procedure check_intersection to detect whether they intersect each other.

7. Syscall Services

We have implemented a group of additional syscall services to support game related functions (e.g. teachers' movements). For your code to work, you should use the modified MARS (`NewMars.jar`) provided in the project section of our course website. Note that not all the new syscalls are necessary to complete your work, some are described here for you to understand the skeleton.

Syscall code should be passed to `$v0` before usage.

Service	**Code**	**Parameters**	**Result**
Create the game screen	200		
Refresh the game screen	201		
Play game sound or stop playing it	202	$a0 = sound ID $a1 = 0: play once; 1: play repeatedly in loop; 2: stop playing The sound IDs are described as follows: 0: the background music; 1: the sound effect of	Play the sound of the given sound ID or stop any playing of it, depending on the given value in $a1.

		obtaining game scores; 2: the sound effect of losing game scores; 3: the sound effect of losing the game; 4: the sound effect of passing a game level; 5: the sound effect of ending Shield mode.	
Set the game score	203	\$a0 = new game score	
Set the game level	204	\$a0 = new level no.	
Create an object	205	\$a0 = unique ID of the object \$a1 = x-coordinate \$a2 = y-coordinate \$a3 = object type: 0 for score point; 1 for student; 2 for teacher	A new object of the given ID and object type is created, replacing any existing object of the same ID and object type. The location of the object is set to the given x- and y-coordinates.
Set the location of an object	206	\$a0 = ID of the object \$a1 = x-coordinate \$a2 = y-coordinate \$a3 = the object's type: 0 for score point; 1 for student	The location of the object is set to the given x- and y-coordinates.
Create a Text object	207	\$a0 = unique ID of the object \$a1 = x-coordinate \$a2 = y-coordinate \$a3 = base address of a text string	Create an object for displaying the text string at the given x- and y-coordinates.
Get the location of a teacher object	208	\$a0 = ID of the object	\$v0 = x-coordinate of the location. \$v1 = y-coordinate of the location.
Get a random path	209		\$v0 = x-coordinate (at the top-left

cell of the maze			corner) of the cell. $v1 = y-coordinate of the cell.
Get a random duration for a score point object	210		$v0 = random number of game iterations for the duration of the visible or hidden status of a score point object
Update the Quiz mode	211	$a0 = 0 for ending the mode; 1 for triggering the mode; 2 for updating the mode's display near its expiration	$v0 = choice no. of the quiz's correct answer choice.
Update the Shield mode	212	$a0 = 0 for ending the mode; 1 for triggering the mode; 2 for updating the mode's display near its expiration	
Move teachers	213		Move all the teacher objects for one game iteration.

8. Submission

You should *ONLY* submit the file comp2611_project_yourstudentid.s with your completed code for the project. Please write down your name, student ID, and email address (as code comments) at the beginning of the file.

Submission is via CASS. The deadline is a hard deadline. Try to avoid uploading in the last minute. If you upload multiple times, we will grade the latest version by default.

9. Grading

Your project will be graded on the basis of the functionality listed in the project description and requirements. You should ensure that your completed program can run properly in our modified MARS. It must also work in a computer in our lab room 4213.