**Project 6 Grading Guide**

|  |  |  |  |
| --- | --- | --- | --- |
| **Item no.** | **Description** | **Correctness** | **Style** |
| PROBLEM 1 | rectArray.m | Total = 9 | Total = 6 |
| 1 | Correctly generates the boundaries for the rectangles using xmax and ymax. (Eg. uses rand to generate points between 0 and xmax, 0 and ymax). | 1 |  |
| 2 | Generates the correct number of rectangles (n rectangles). | 1 |  |
| 3 | Uses a for loop to generate the rectangles. |  | 1 |
| 4 | Returns a struct array. | 1 |  |
|  | rectangles.m |  |  |
| 5 | For Figure 1, calls ShowRect on all rectangles in Z. | 1 |  |
| 6 | For Figure 1, chooses a single color for all rectangles in Z. |  | 1 |
| 7 | For Figure 2, correctly calculates the area of the rectangles. | 1 |  |
| 8 | For Figure 2, uses the built­in sort function given in the project assignment [y, idx] = sort(x) . |  | 1 |
| 9 | For Figure 2, correctly colors the rectangles by linearly interpolating: f \* colorL + (1­-f) \* colorS . | 1 |  |
| 10 | For Figure 2, correctly calculates f as (area­-minarea)/(maxarea-­minarea) . | 1 |  |
| 11 | For Figure 2, returns colrZ as an n-by-­3 matrix where colrZ(k,:) are the rgb values of Z(k). | 1 |  |
| 12 | For Figure 2, returns colrSorted as an n-­by-­3 matrix where colrSorted(i,:) are the rgb values of the ith smallest rectangle. | 1 |  |
| 13 | For Figure 2, uses vectorized code for color interpolation. |  | 1 |
| 14 | For Figure 2, label the figure with “Biggest” and “Smallest”. |  | 1 |
| 15 | For both Figure 1 and Figure 2, add a descriptive title. |  | 1 |
| PROBLEM 2 | Event.m | Total = 21 | Total = 10 |
| Note | Students may use either “fill” or “drawRect” for this problem. |  |  |
| 16 | Correctly calculates the earliestTime by determining the overlap between available and possibleInterval and then making sure the overlap is larger than duration. | 1 |  |
| 17 | Uses either the overlap method or calculates the points leftoverlap and rightoverlap. Does NOT use an extremely long if/elseif/else statement. Also does NOT break when an empty interval is passed in. |  | 2 |
| 18 | Correctly fills in setScheduledTime, unschedule, and getId. | 1 |  |
| 19 | Draws the black border for an event regardless of whether or not it’s scheduled (1C) at the correct x/y coordinates (1C). | 2 |  |
| 20 | Draws a colored rectangle for the event if it is scheduled. | 1 |  |
| 21 | Correctly colors the rectangle by the following formula:  self.importance \* bestColor + (1 - ­self.importance) \* worstColor | 1 |  |
| 22 | The height of the rectangle is less than (or equal to) 1. |  | 1 |
|  | Course.m |  |  |
| 23 | Correctly implements the getCourseName function. | 1 |  |
| 24 | Calls Event’s draw method using draw@Event(self). | 1 |  |
| 25 | Adds the id and course name to the middle of the bar if the course is scheduled (1c). Text should be horizontally aligned in the center as shown in solution. | 1 | 1 |
|  | Schedule.m |  |  |
| 26 | Correctly implements the constructor and addEvent. |  | 1 |
| 27 | In scheduleEvents, does not change any properties in Schedule object. |  | 1 |
| 28 | In scheduleEvents, correctly loops over the events and unschedules them. | 1 |  |
| 29 | In scheduleEvents, finds the best event ordered by earliest time (and then importance/duration ratio). | 1 |  |
| 30 | In scheduleEvents, uses the function earliestTime when finding the best event. |  | 1 |
| 31 | In scheduleEvents, schedules the best event found (1C) and then updates the remaining window (1C). | 2 |  |
| 32 | In scheduleEvents, uses a while loop for this method with a correct termination condition. |  | 1 |
|  | In scheduleEvents, return value is a cell array of the Events (and Courses) that did not get scheduled. | 1 |  |
| 33 | In draw, draws all the events (1c) and sets the axes (1s). | 1 | 1 |
|  | createSchedule.m |  |  |
| 34 | Reads the event file correctly. | 1 |  |
| 35 | Add event to s. | 1 |  |
| 36 | Checks whether an each event is a Course. | 1 |  |
| 37 | Calls scheduleEvents() and draws when appropriate. | 1 |  |
|  | testScript.m |  |  |
| 38 | Script runs without breaking. | 2 |  |
| 39 | Script covers more than 50% of the implemented code. |  | 1 |
| PROBLEM 3 | sky.m | Total = 9 | Total = 1 |
| NOTE | Due to a typo in the assignment, we accept functions named EITHER sky or mySky. A function named mySky won’t run correctly since CMS renames the file to sky.m. Do not deduct points for this! |  |  |
| 40 | Random noise is computed correctly from a normal distribution. | 1 |  |
| 41 | The center color is the average of the corners plus the random noise. | 1 |  |
| 42 | Color components are always between 0 and 1. | 1 |  |
| 43 | Coordinates are computed correctly, and (x,y) is treated as the center of the rectangle. | 1 |  |
| 44 | Base case is taken when either w or h is less than minsize (1C), and fills a rectangle with the correct color (1C). | 2 |  |
| 45 | Calculates the colors at the midpoint of each side (1C). Assigns them meaningful variable names (acronyms are OK) (1S). | 1 | 1 |
| 46 | Four correct recursive calls (-1 for each unique error, -2 max). | 2 |  |
| GENERAL |  |  | Total = 10 |
| G1 | Script starts with a concise comment describing the program.  Function comment follows function header. |  | 1 |
| G2 | Code is sufficiently (but not excessively) commented. |  | 1 |
| G3 | Line lengths are not excessively long (80 columns).  NOTE: It's ok if a couple lines are a little too long, especially if they are print statements |  | 1 |
| G4 | No extra output (debugging output) produced |  | 1 |
| G5 | Proper indentation is always used. |  | 1 |
| G6 | Use meaningful variable names. Do not overwrite MATLAB keywords. **Note:** For this assignment, it is OK to overwrite the keyword “beta”. |  | 1 |
| G7 | Name important parameters as variables (constants). |  | 1 |
| G8 | No superfluous code (e.g., an empty if or else branch or a useless loop). Of course some students will have code that is awkward or unclear or inefficient. This point is specifically for not having code that does literally nothing. |  | 1 |
| G9 | Reasonably efficient code. |  | 1 |
| G10 | Does NOT put semicolon at wrong places, e.g., at the end of these lines: "if", "elseif", "else"," for","while", "function". |  | 1 |
| TOTAL |  | 39 | 27 |

**Penalties**

|  |  |  |
| --- | --- | --- |
| P1 | Student's code does not execute (or student provides a script when a function is required and vice-versa) | -1 from final score |
| P2 | Student's code crashes or does not terminate (infinite loop) for normal cases. | -1 from final score |
| P3 | All function headers and file names match those specified in the project description exactly. All input and output variables should be of the correct type. | -1 from final score |

**Grade Calculation**

|  |  |
| --- | --- |
| Total Possible Correctness Points | TC = 39 |
| Total Possible Style Points | TS = 27 |
| Student Correctness Points | C = min( \_\_\_ + 1 freebie point, TC) |
| Student Style Points | S = min( \_\_\_ + 1 freebie point, TS) |

Exceptions: If any file is missing/unacceptable, no freebie points can be applied to that file and subtract 3 style points for each missing/unacceptable file.

|  |  |
| --- | --- |
| Student’s final score | ([(C/(TC))+(S/TS)] X 5) - Penalties  (Out of 10; 1 decimal; no negative score; round to NEAREST) |