NAME _____ STUDENT ID: __ __ __ __ __ __ __ __

University of Waterloo

# CS 349 Fall 2018 Midterm Exam

Wed Oct 24, 2018 from 7:00 – 8:50 PM

Instructions

- Write your full name at the top of every page.
- This is a 110-minute closed book exam. No calculators or aids are permitted.
- Proctors will not interpret questions, they can only confirm or deny errors.
- If you consider a question ambiguous, state your assumptions and answer to the best of your ability.

## Instructions

1. Verify you have one exam booklet with 12 pages.
2. All solutions must be placed in this booklet.
3. Proctors will only answer two questions: "Is this a typo?" and "What is the meaning of this non-technical English word?"

## Standard Definitions

- T(tx, ty), R(Θ), and S(sx, sy) represent 2D affine transformation matrices for translation, rotation, and scaling respectively. The identity matrix is represented as I.
- All rotation angles are in degrees, unless stated otherwise.
- All coordinates are left-handed window coordinates (origin at top left), unless stated otherwise.
- In window coordinates, positive Θ rotates clockwise.
- The rotation transform is: $x' = x\cos\Theta - y\sin\Theta$, $y' = x\sin\Theta + y\cos\Theta$

| Part | Marks |
|---|---|
| 1. Short answer | 20 |
| 2. Windowing | 8 |
| 3. Events | 16 |
| 4. MVC | 10 |
| 5. Layout | 8 |
| 6. Graphics | 16 |
| Total | 78 |

## 1. SHORT ANSWER (20 marks)

(a) We discussed three styles of interaction in the history of computing: batch interfaces, conversational interfaces, and graphical user interfaces. Name two advantages of a graphical user interface, compared to a ~~batch or~~ conversational interface [2 marks]

Any two:

- Promotes recognition over recall (i.e. users can recognize features)
- GUI interface is explorable
- No commands to memorize
- Encourages novice use
- Supports event-driven programming – not really specific to GUI but implied

(b) What is the difference between the terms *interface* and *interaction*? [2 marks]

Interface is what the user sees (output) and manipulates (input). Can be text-driven, or graphical.

Interaction is the process of providing input and getting feedback through the interface over a period of time, or while performing a specific task (i.e. it's iterative).

(c) The Base Window System (BWS) enables applications to use a canvas abstraction in their window. What is a canvas abstraction and why is it useful? [2 marks]

The canvas is the part of the area that the application exclusively "owns" (vs. BWS that owns the window, and WM that owns the decorations). This helps the BWS in: dispatching events, drawing windows, or keeping applications secure/memory separate (any one is fine).

(d) What is a *graphics context*? Why is it useful? [2 marks]

A graphics context is a structure (or class) containing a set of common drawing properties. This allows us to define a palette (brush, context) to use when drawing, so that we can set all required properties on one action (or makes it easy to set defaults).

(e) Give an example where pure positional dispatch would fail. What is a solution? [2 marks]

Any instance where we want to send events to a widget that isn't under the cursor e.g. scrollbar example in class where we move the mouse off of the scrollbar, but it continues to receive events; mouse_down over a button, move away then mouse_up.

The solution is to use focus dispatch to force events to dispatch to a widget.

(f) Is the transformation `T(4,5)·T(1,2)` the same as `T(1,2)·T(4,5)`? Explain. [2 marks]

Yes because they're both translations, which produce the same results in either order (you can also show mathematically, but not required)

Order typically matters with mixed operations, or scaling/rotation, but doesn't matter in this particular case.

No marks for saying that multiplication is not commutative.

(g) What is the exact 3x3 transformation matrix represented by `scale (3,4)`? [2 marks]

```
3   0   0
0   4   0
0   0   1
```

(h) What 2D points do these homogeneous coordinates represent? [3 marks]

$$A = \begin{bmatrix} 4 \\ 8 \\ 0.5 \end{bmatrix} \quad B = \begin{bmatrix} 15 \\ 9 \\ 0 \end{bmatrix} \quad C = \begin{bmatrix} -3 \\ -6 \\ -1 \end{bmatrix}$$

A = (8,16)
B = not a point, vector (w=0)
C = (3,6)

(i) What fields would you use to represent a *rectangle shape model?* [2 marks]

Any two from:

- Position x, y
- Width, Height
- Colour
- Border style or thickness
- Something else reasonable

(j) Name one *UI invention* introduced in Douglas Englebart's "On-Line System (NLS)". [1 mark]

Mouse, or chording keyboard, or remote/collaborative work (or hypertext).

## 2. WINDOWING SYSTEMS (8 marks)

(a) Describe the roles of the base window system and window manager in X Windows. What specifically are the responsibilities of each of these subsystems? (4 marks)

The base window system is responsible for creating windows, and managing low level events: packaging them in a common structure and dispatching to the application window, or widget (for HW toolkit).

The window manager is responsible for window decorations - "look and feel" of the window, plus any widget directly attached to the window frame (min, max buttons, resize widgets).

(b) For each of the following statements, indicate whether it's True (T) or False (F). (4 marks in total)

__F___ The Base Window System is multi-threaded to handle multiple windows. – always single threaded

__F___ The window manager is a separate application that owns the title bar and decorations of each window; the Base Window System owns everything else. – application owns the canvas

__T___ X Windows encourages programmers to select which events their programs receive with a set of predefined flags that are ORed together.

__T___ A graphics context includes settings related to background colours, line widths, line joining policies, and how newly drawn bits are combined with bits already on the screen.

## 3. EVENTS (16 marks)

For questions (a), (b) and (c), consider the following Xlib code fragment:

```
// time of the last frame (in milliseconds)
long lastTime = 0;

while(true) {

   // gets current time in milliseconds
   long t = getTimeMs();

   if ((t - lastTime) > 100) {
      lastTime = t;

      // update animated part of scene model using current time
      update_animated(t);

      if (XPending(display) == 1) {
         XEvent event;
         XNextEvent(display, &event);
         // process event and update scene model
         handle_event(event);
      }
   }

   // repaint the scene using updated scene model
   paint_scene();
}
```

(a) There is at least one *resource usage issue* with this code. What is it, and how can it be fixed? [2 marks]

Two possible answers

1 ISSUE: scene is painted much too often since scene can only change every 100ms
1 FIX: could fix by moving paint_scene inside the if condition

or

1 ISSUE scene is painted much too often  AND/OR loop executes more often than needed ( … using up CPU and battery);
1 FIX add a sleep

(b) The *event processing* part of this code has two bugs. Identify them, explain why each is a bug, and how each could be fixed. [4 marks]

Problem A
1 it won't process any events if more than two are in the queue
1 change  xpending == 1 to xpending > 0

Problem B
1 it only processes one event every 100ms, many events can be faster than that
1 change if pending to while pending

(c) Why would the programmer use two different functions to animate and paint the scene? [2 marks]

A few possible answers:

- may want to paint without animating (e.g. window resize) – main reason
- separate paint function lets developer use painters algorithm more easily
- paint function uses a display list and Displayables (must mention these terms)

Also accept generic software engineering reasons if they're clear, like:
- decoupling code (good OO design) by separating painting from state changes for animating

Parts (d) and (e) should be interpreted with the following assumptions:

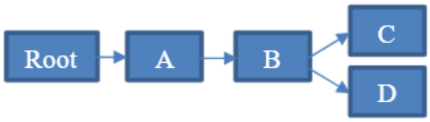
- The coordinate system places the origin in the top-left corner of the window (x grows as you move right, and y grows as you move down the screen).
- All component locations are specified relative to the parent component.
- All components are visible.

Given:

- The screen's resolution is 1280x1024.
- Component A is a window, located on the screen at (10, 10) with a width of 500 and height of 500.
- Component B is a child of component A. It has a location of (15, 10), width 100 and height 100.
- Component C is a child of component B. It has a location of (25, 25), width 25 and height 40.
- Component D is a child of component B. It has a location of (55,25), width 10 and height 40.

Answer the following questions:

(d).Draw the interactor tree for these components, and label each component by name. Clearly label the root of the tree. [4 marks]




It's also fine to label A as the root..

(e) Given the specifications above, suppose the user on the screen at position (75, 50) (i.e. in screen coordinates). List each component that will receive the event, and give the coordinates of the mouse event when the component receives it. [4 marks]

Relative to A, (75,50) has coordinates (75,50) - (10,10) = (65,40) (i.e. coordinates local to A's coordinate system). A has width 500, height 500, so this point falls within A's boundaries, and is a hit.

Relative to B, point (65,40) is has coordinates (65,40) - (15,10) = (50,30). B has width 100, height 100 so this is a hit.

Relative to C, point (50,30) has coordinates (50,30) – (25,25) = (25,5). C has width 25, height 40 so this is a hit.

Relative to D, point (50,30) has coordinates (50,30) – (55,25) = (-2,5). D doesn't get hit.

## 4. MVC (10 marks)

(a) Model-View-Controller (MVC) is a common design pattern used in UI toolkits. Define each term (Model, View, and Controller) and its role or function in the overall system. [6 marks]

Model: Internal state or data for the application. Loosely coupled to view/controller; notifies views when the state changes.

View: The interface that the user sees, displaying output. Loosely coupled to the model; receives updates from the model determining when it needs to paint or refresh itself, and fetches data from the model.

Controller: Mediates user/system by managing all user input; interprets input and passes events (or calls methods, or changes states in) the model.

(b) When implementing MVC, you typically instantiate the *model* and *view* in order (below). Why are they instantiated in this order, and why is the model passed as an argument to the view? [2 marks]

```
Model m = new model();
View v = new view(m);
```

Model must exist before creating the view. The model is independent and can be created independently of any views. The view needs to know how to call the model to fetch data, so the model must be passed as a reference.

(c) When implementing MVC, the *view* and *controller* are often combined into a single class. Why is this done? [2 marks]

When using widget toolkits, the widgets (and view) are used for both input and output i.e. the functionality is tightly coupled. We typically implement the controller as a listener on the widgets, and keep them contained in the view. (Also correct to talk about controller/view interaction where widget updated may not get sent to the model – but this is a little esoteric).

## 5. LAYOUT (8 marks)

(a) Fixed layout refers to a widget layout strategy in which the size and location of widgets are hardcoded. This is rarely used – instead we prefer to use dynamic layout strategies that adapt to changes in window size. Name and describe a general layout strategy where the size of widgets is determined dynamically. Also provide an example of a Java layout manager that uses this strategy. Although not required, you may draw a picture if that is helpful. [4 marks]

Intrinsic: the container widget queries its child widgets for preferred size, and attempts to accomodate them (by resizing, repositioning).
e.g. BoxLayout, or FlowLayout
Example must match explanation

Variable intrinsic: as *intrinsic*, but the container widget can make a second pass to adjust itself if required to make it work, and is much more flexible/robost than intrinsic.
e.g. Grid, GridBagLayout, or BorderLayout
Example must match explanation

Struts and Springs: specify rigid/expanding areas, widgets can expand to fill.
e.g. SpringLayout, or BoxLayout
Example matches explanation

(b) Although we rarely use fixed layouts, there are times where they might be an acceptable choice. Describe the problem with fixed layouts (i.e. why we avoid them), and a specific situation where the choice of a fixed layout might be appropriate. [4 marks]

We avoid fixed layouts because they cannot adapt to changes in window size or screen. They can be an issue anytime the contents may not fix the fixed window. e.g. a dialog where the contents may change (English to German for instance).

They might be suitable when you know that the content will always fit or that the resolution will never change. e.g. kiosk, phone that only supports full-screen apps of a single resolution.

## 6. GRAPHICS (16 marks)

(a) Your task is to write code to render each picture, using no more than 8 function calls. Assume each picture is initially white. Although the canvas is a 100x100 grid, you may draw outside of these boundaries (although anything drawn outside the boundaries will not be shown). The overlaid grid is provided for your reference only, it is not part of the drawing.

You may only call this function:

```
// draw an s by s square filled with colour c with top left at (x,y)
// Colour c can be only WHITE, BLACK, GREY
square(Colour c, int x, int y, int s) { ... }
```

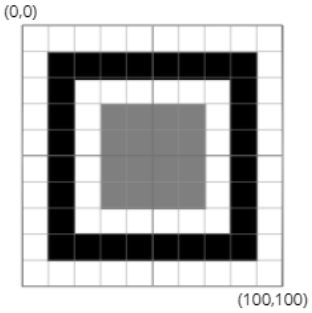
Here's an example call:

```
// draw a 30 by 30 black square with top-left corner at (10,20)
square(BLACK, 10, 20, 30)
```

There may be multiple solutions. Any solution that works with 8 or fewer calls is acceptable. No solution will be accepted that uses other functions, or language constructs.

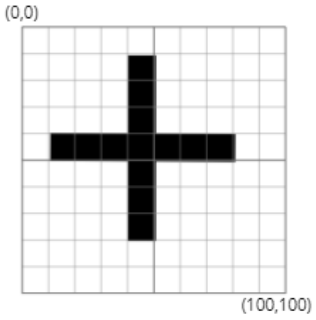
(a) Write code to render the image at right. [3 marks]

```
blackRect(10,10,80)
whiteRect(20,20,60)
grayRect(30,30,40)
```



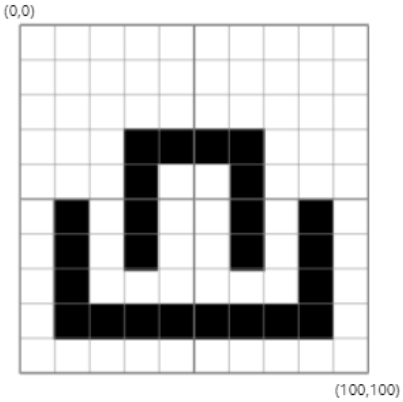

(b) Write code to render the image at right. [3 marks]

```
black(10,10,70) // or black(10,10,40),
black(40,10,40)
white(10,10,30) // or white(0,0,40)
white(50,10,30) // or white(50,0,40)
white(10,50,30) // or white(0,50,40)
white(50,50,30) // or white(50,50,40) or
white(50,50,50)
```



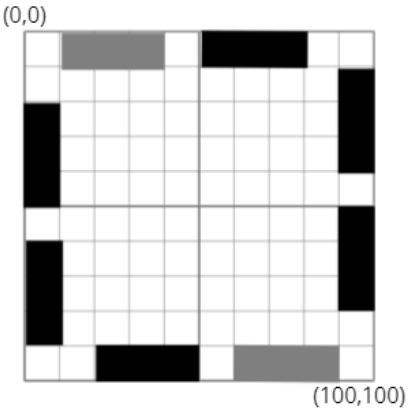
(c) Write code to render the image at right. [3 marks]

```
black(10,50,40)
white(20,50,30)
black(50,50,40)
white(50,50,30)
black(30,30,40)
white(40,40,20)
white(40,50,20) // overlaps with above
```




(d) Write code to render the image at right. [3 marks]

```
gray(10,-20,30)  // top
black(50,-20,30)
black(-20,20,30) // left
black(-20,60,30)
black(90,10,30)  // right
black(90,50,30)
black(20,90,30)  // bottom
gray(60,90,30)
```



I have an irregular shape (2 x 2), shown in Figure 1, whose lower-left corner is at (5, 4).

I want to produce the results shown in Figure 2, where the shape has been rotated by 90 degrees around it's center.

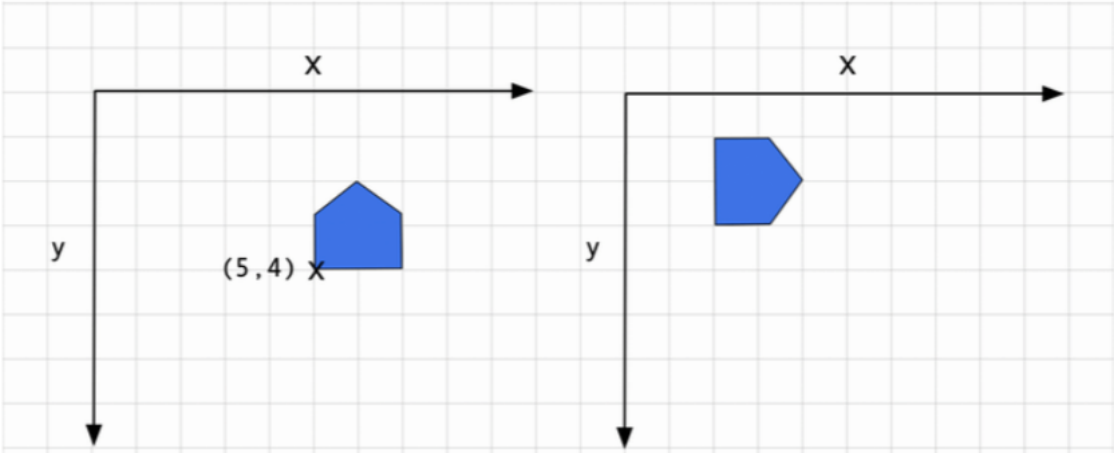



Figure 1: Original

Figure 2: Transformed

The following Java functions are available:

```
scale(x,y);          // scales non-uniformly by x and y values
rotate(theta);       // rotate by theta degrees clockwise
translate(dx,dy);    // translate by dx and dy
drawShape();         // draw this shape model
```

(c) Using *only* these functions, write code to transform the shape from `Figure 1`, to produce the results shown in `Figure 2`. [4 marks]

```
T(-6,-3) // rotate around centre
R(90)
T(3,2)
drawShape()
```

or

```
T(-5,-4) // rotate around bottom left
R(90)
T(2,1)
drawShape()
```