

# INTERACTING WITH GRAPHICS

CMPT 381

# Outline

---

- Working with paths
  - Bounds, Distance to object, Nearest point, Object intersection, Inside/Outside
- Transformations
  - Translation, Scaling, Rotation
- User interactions
  - Handles, Object creation, Selection, Transformation, Snapping, Grouping

# Doing it yourself

---

- In an application-defined graphical workspace:
  - There are no UI layers to help you determine:
    - Which object was clicked on
      - “hit detection”
    - Where objects should be drawn
      - e.g. arrows between objects
- What do toolkits provide?
  - Canvas:
    - Inside/outside, bounding box
    - Translate, Scale
  - Java Shape API:
    - Inside/outside, bounding box, intersection check
    - Translate

# Path equations

---

- Bounds
- Distance from a point to an object
- Nearest point on an object
- Object intersection
- Inside/Outside

# Bounding box

---

- Why do we need bounding boxes?
  - Checking inside/outside of a rectangle is easy
- If convex hull property true:
  - Use the min and max of the control points
  - For a line:
    - $\text{left} = \min(x_1, x_2)$
    - $\text{right} = \max(x_1, x_2)$
    - $\text{top} = \min(y_1, y_2)$
    - $\text{bottom} = \max(y_1, y_2)$
- If not true?
  - E.g. circle

# Distance from point to object

---

- Why:
  - Is a mouse click close enough to select the object?
- Use implicit equation for the object
  - for a line:
    - $\text{Line}(x,y) = Ax + By + C$
    - For points on the line, returns 0
    - For points left/above the line, returns negative
    - For points right/below the line, returns positive
  - for a circle:
    - $\text{Dist}(x,y) = \sqrt{(x - x_C)^2 + (y - y_C)^2} - R$

# Nearest point on an object

---

- Why?
  - E.g. drawing arrows between objects
- Use implicit equation again
  - For line:
    - $\text{Nearest}(x) = x - A * \text{Line}(x,y)$
    - $\text{Nearest}(y) = y - B * \text{Line}(x,y)$

# Intersection of objects

---

- Why
  - Not frequent, but sometimes important
- Lines
  - Linear equations
  - Algebraic
- Circles, arcs, ellipses
  - Quadratic equations
  - Algebraic
- Rotated ellipses, splines, curves
  - Difficult
  - requires numerical methods



# Inside/Outside

---

- For selection of filled objects
- For general shapes
  - inside/outside is difficult
  - Use bounding boxes first
  - Determine which objects must be checked in more detail
- For simple shapes
  - use basic equations
- Java Graphics2D Shape API

# Transformations

---

- Translation
- Scaling
- Rotation
- Each transformation uses a pair of equations
  - maps  $(x,y)$  to  $(x',y')$
- Each transformation uses coefficients to determine magnitude of the change
  - Amount of movement, scaling, or rotation
- Matrix versions for these (and more) transformations

# Translation

---

- Moves an object from one location to another
- Adds a distance to  $x$  and  $y$

$$x' = x + dx$$

$$y' = y + dy$$

# Scaling

---

- Changes the size of an object
- Multiplies  $x$  and  $y$  by scale factor

$$x' = x * s_x$$

$$y' = y * s_y$$

- Note
  - Scaling is done relative to the origin
    - Objects not at the origin will move as well
  - What does a negative scale factor do?

# Rotation

---

- Rotates an object counterclockwise
- Adds a distance to x and y

$$x' = \cos(a) * x - \sin(a) * y$$

$$y' = \sin(a) * x + \cos(a) * y$$

- Note
  - Again, rotates around the origin

# Combining transformations

---

- Often need multiple transformations
  - E.g. in-place scaling/rotation
    - Translate to origin
    - Scale or rotate
    - Translate back to starting position
- Homogeneous coordinates
  - Represent all transformations in similar format
  - There are matrix representations for each transform
  - Matrix multiplication is the combination mechanism
    - Also allows undo of any transformation

# User interactions

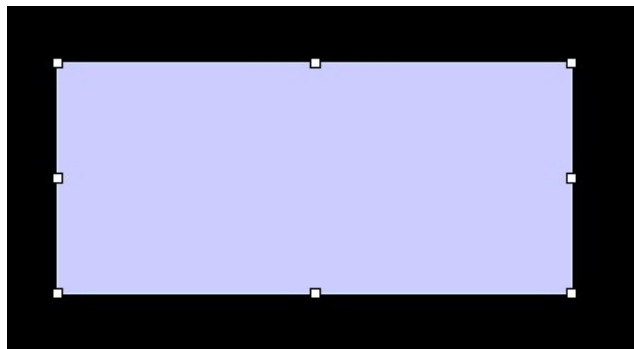
---

- Handles
- Interactive object creation
- Selection
- User-controlled transformation
- Snapping
- Grouping

# Handles

---

- Basic interaction paradigm
  - Select-and-manipulate
- MVC view of graphical shapes
  - Model stores control points of the shape
  - View can make control points visible to user
    - “selection handles”
- Allows for interactive manipulation





# Interactive object creation

---

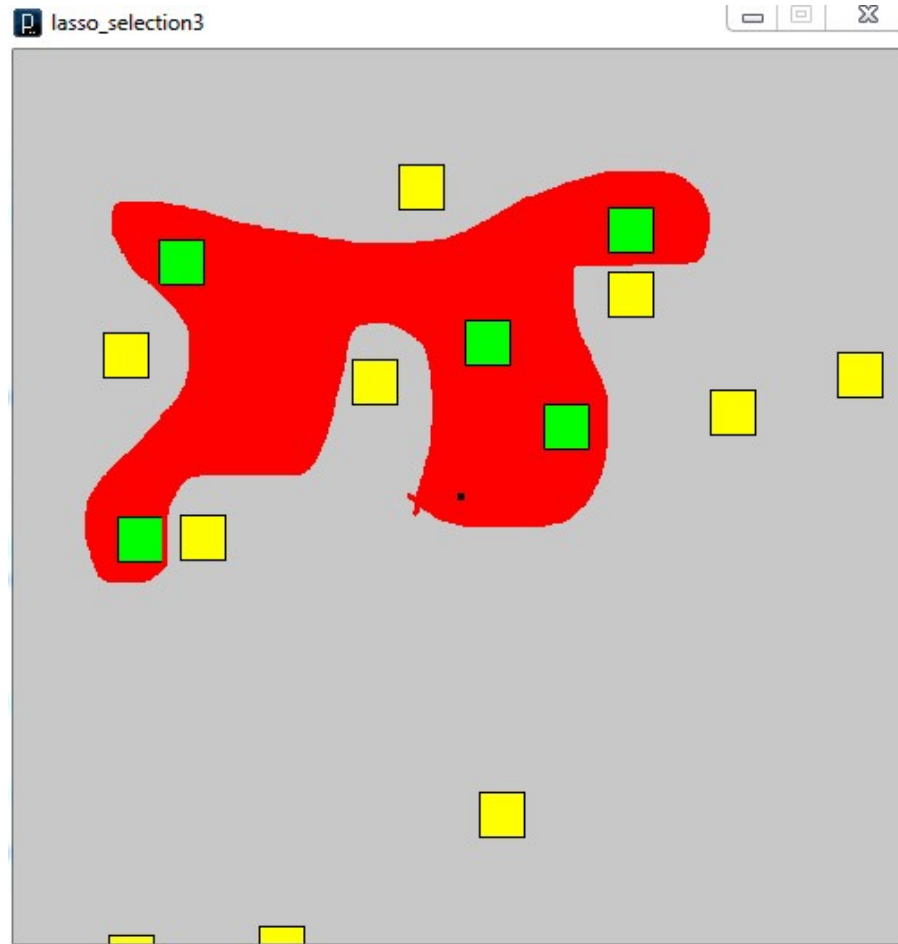
- Provide feedback during creation
  - Improves accuracy and provides preview
- Two approaches
  - If object does not depend on other objects
    - E.g. node in node-and-edge editor
    - Create object on initial mouseDown
    - Then proceed as if resizing
  - If object has dependencies
    - E.g. edge in node-and-edge-editor
    - Must create temporary representation for the object
    - Then check validity on mouseUp
    - Note:
      - Where in MVC is this temporary object kept?

# Selection

---

- Traditional interaction rules:
  - mouseDown selects
  - mouseDown+mouseMove drags
  - an object stays selected until:
    - Another object is selected
    - mouseDown on no object occurs
- Multiple selection:
  - Discrete:
    - Add objects to the selected list with modifier key
  - “Rubber-band” area:
    - Initiated by clicking on background and dragging
    - Requires visual feedback
    - Requires intersection testing on mouseUp

# Lasso selection



# User-controlled transformations

---

- Translation
  - Done by dragging on shape area (not handle)
- Scaling
  - Type 1: by dragging control handle
    - Recalculate size based on new control points
  - Type 2: by specific amount (e.g. 200%)
    - Must use scale transformation (in-place)
- Rotation
  - How to differentiate from dragging or scaling
  - What point to rotate around?
    - Again, must translate → rotate → translate

# Snapping

---

- Many graphical tasks involve precise positioning
  - However, difficult to position the mouse cursor accurately
- Snapping
  - Rounds off mouse position to a more appropriate place
  - Snap to grid:
    - Simple localization of position to a rectilinear grid
    - Grid can have arbitrary resolution
    - Some intersections can lie between grid points
  - Snap to object:
    - Localize positions to nearest intersection
    - Uses basic equations discussed earlier

# Grouping

---

- Allows manipulation of composite objects
  - Some objects need grouping for correct transformation
- View organized into a hierarchy
  - Assists with selection, transformation, duplication

