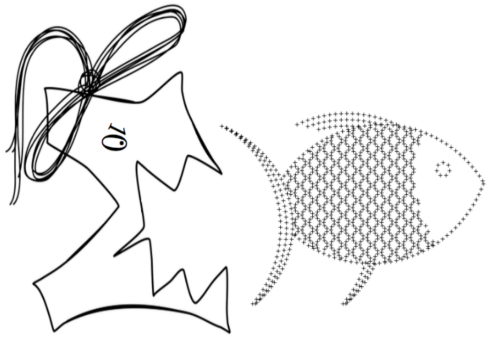
Software:
Tool for Change

Christopher Kumar Anand
McMaster University
with அக்ஷய் குமார் அ



McMaster University
ENGINEERING

From Canada

(my uncle clearing 70cm of snow)

Give Children Tools

- Children are the future
- We want to give them tools to make it better
- Software is the most powerful tool in history
- Each year, we teach 5000
- Need partners to teach 5000000

Today: Elm

- Over 10 years, we have tried many languages
  - Python, Alice, Haskell, …
- Discovered Elm, wrote GraphicSVG
  - Functional (like Haskell)
  - Simple (like Python)
  - Graphical (like Alice)

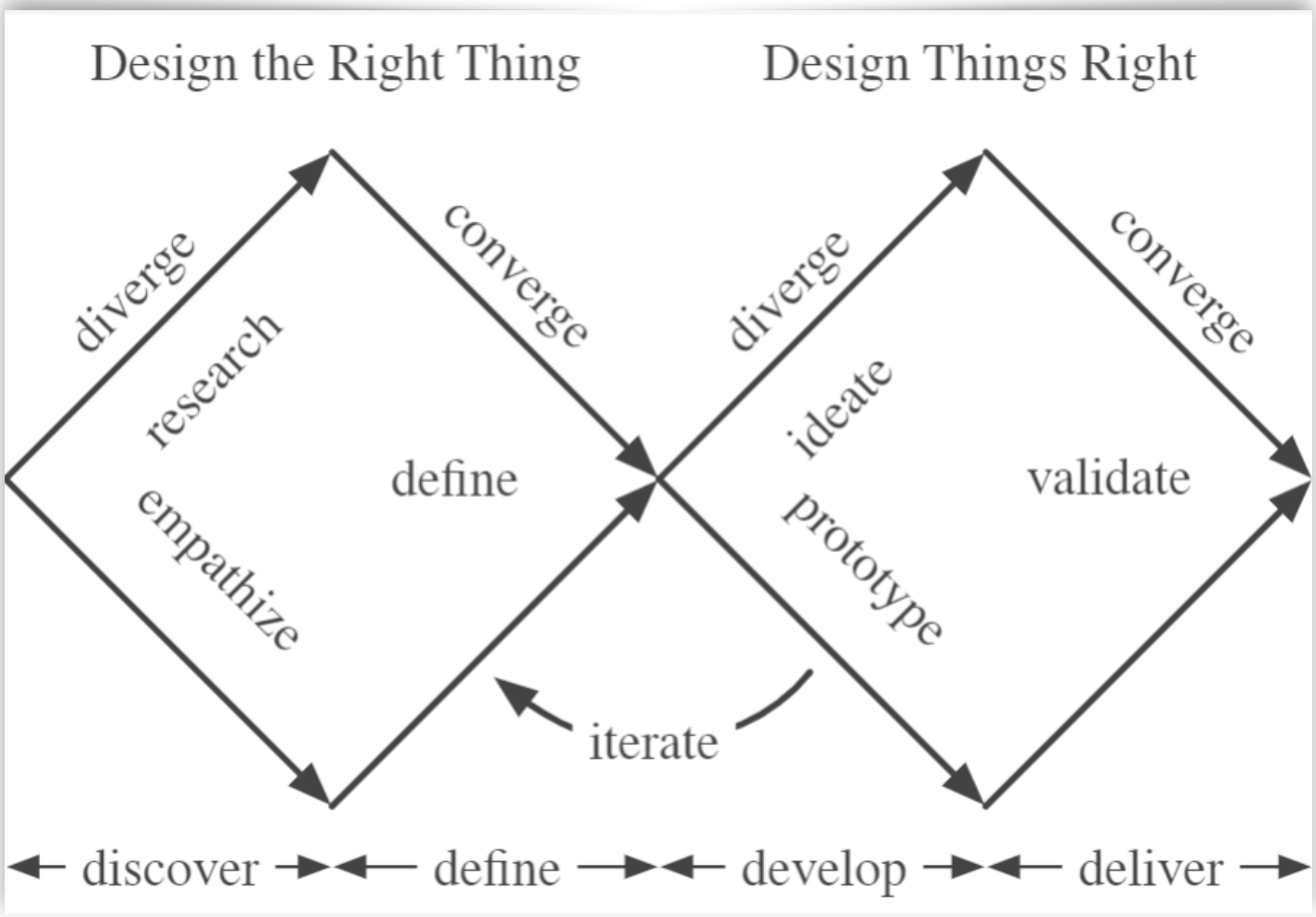
Functional Programming

- Functions are Mathematical Functions
  - Input ➔ Output
  - No side-effects
  - Easy to reason about
    - Practical to prove programs correct
  - Matches semantics of *Algebra*
    - Helps children with toughest concept in mathematics

The Future

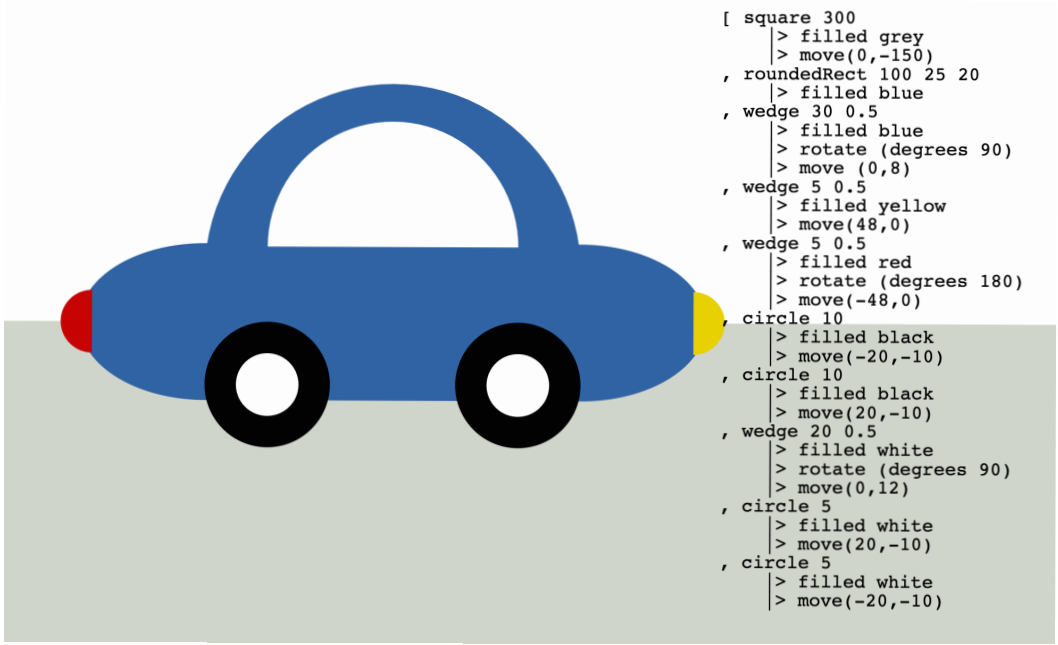
- It is the future of programming
  - Scala adds it to Java
  - Apple integrated it into Swift
  - React imitates it in JavaScript
  - Graduate: “The smartest engineers are the ones learning Haskell.”

Tomorrow: Design Thinking



GraphicSVG

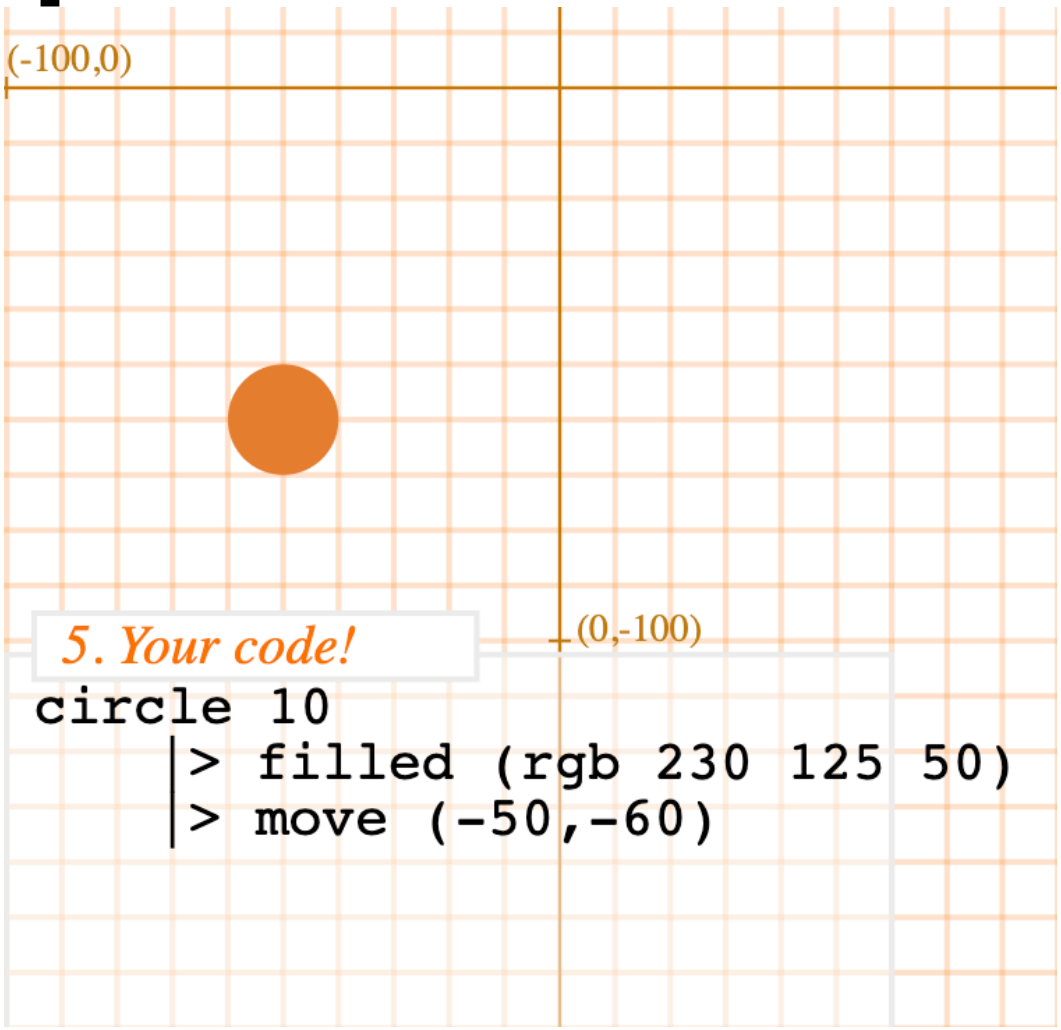
- Graphics library
  - Simple functions
  - Composable functions



```
[ square 300
    |> filled grey
    |> move(0,-150)
, roundedRect 100 25 20
    |> filled blue
, wedge 30 0.5
    |> filled blue
    |> rotate (degrees 90)
    |> move (0,8)
, wedge 5 0.5
    |> filled yellow
    |> move(48,0)
, wedge 5 0.5
    |> filled red
    |> rotate (degrees 180)
    |> move(-48,0)
, circle 10
    |> filled black
    |> move(-20,-10)
, circle 10
    |> filled black
    |> move(20,-10)
, wedge 20 0.5
    |> filled white
    |> rotate (degrees 90)
    |> move(0,12)
, circle 5
    |> filled white
    |> move(20,-10)
, circle 5
    |> filled white
    |> move(-20,-10)
```

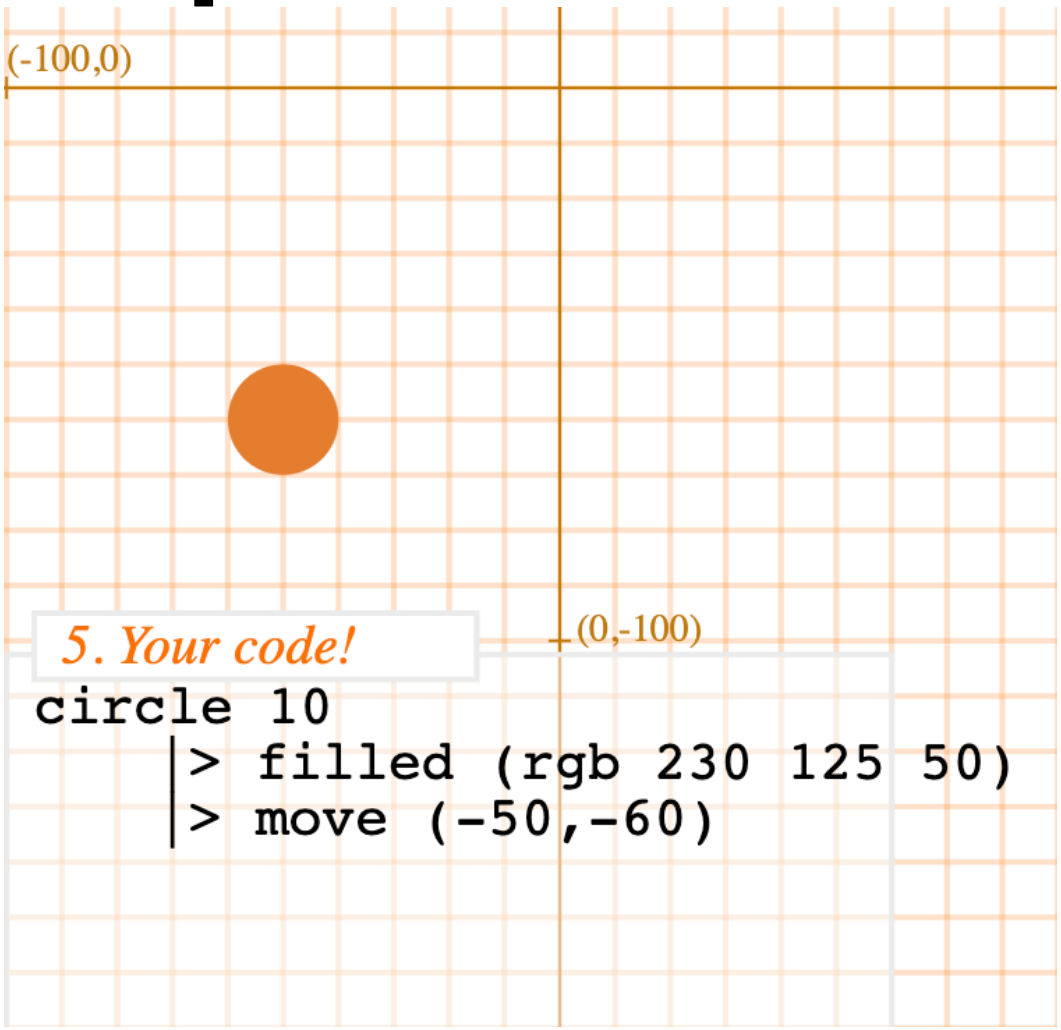
Composable

- `|>`
  - `Composition operation`

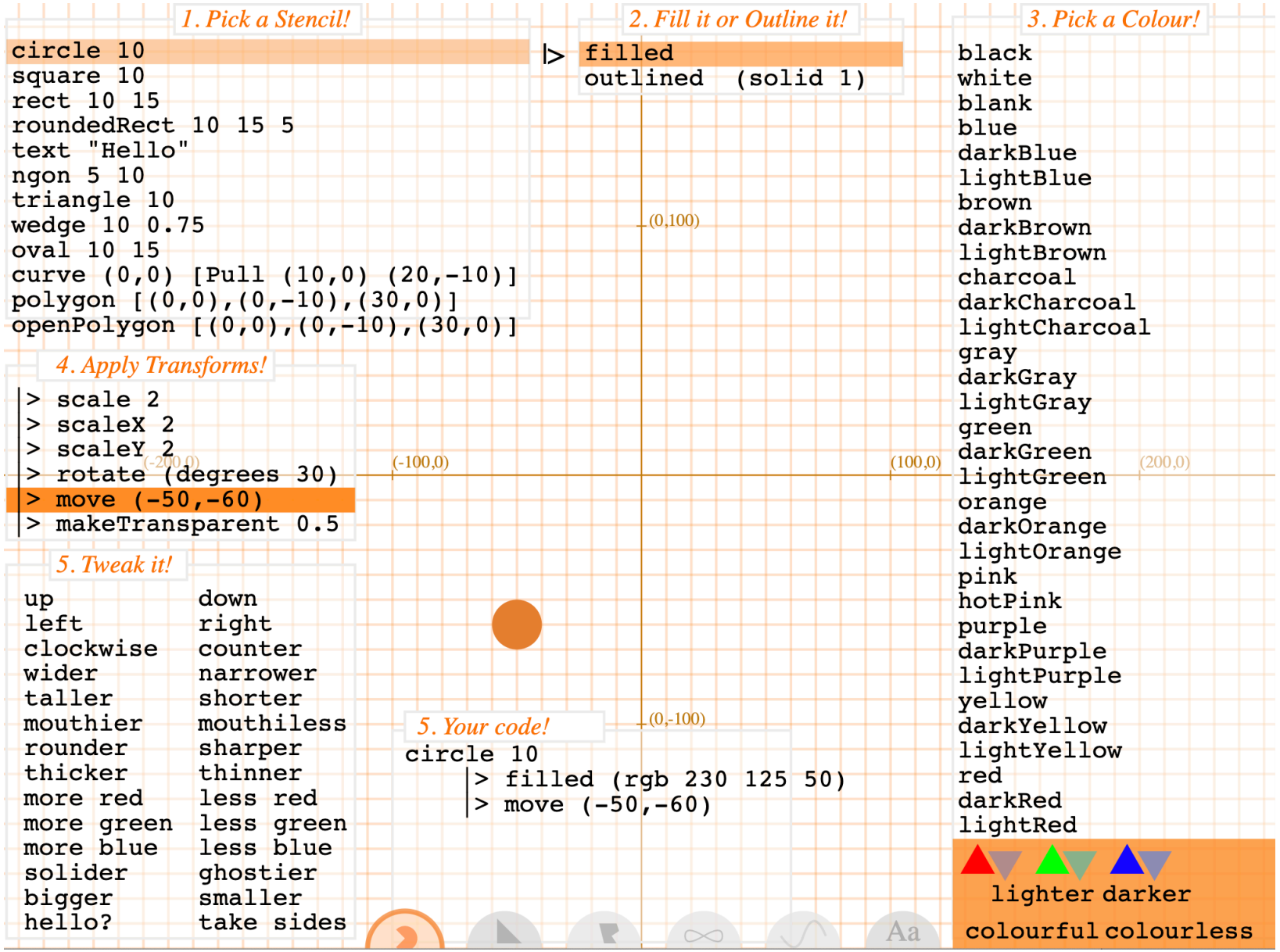


Simple

- `circle` has ***1*** input
- `filled` has ***2*** inputs
  - an rgb colour
  - a Stencil (circle)
- `move` has ***2*** inputs
  - an (x,y) vector
  - a Shape



Everything On One Page



Everything Online

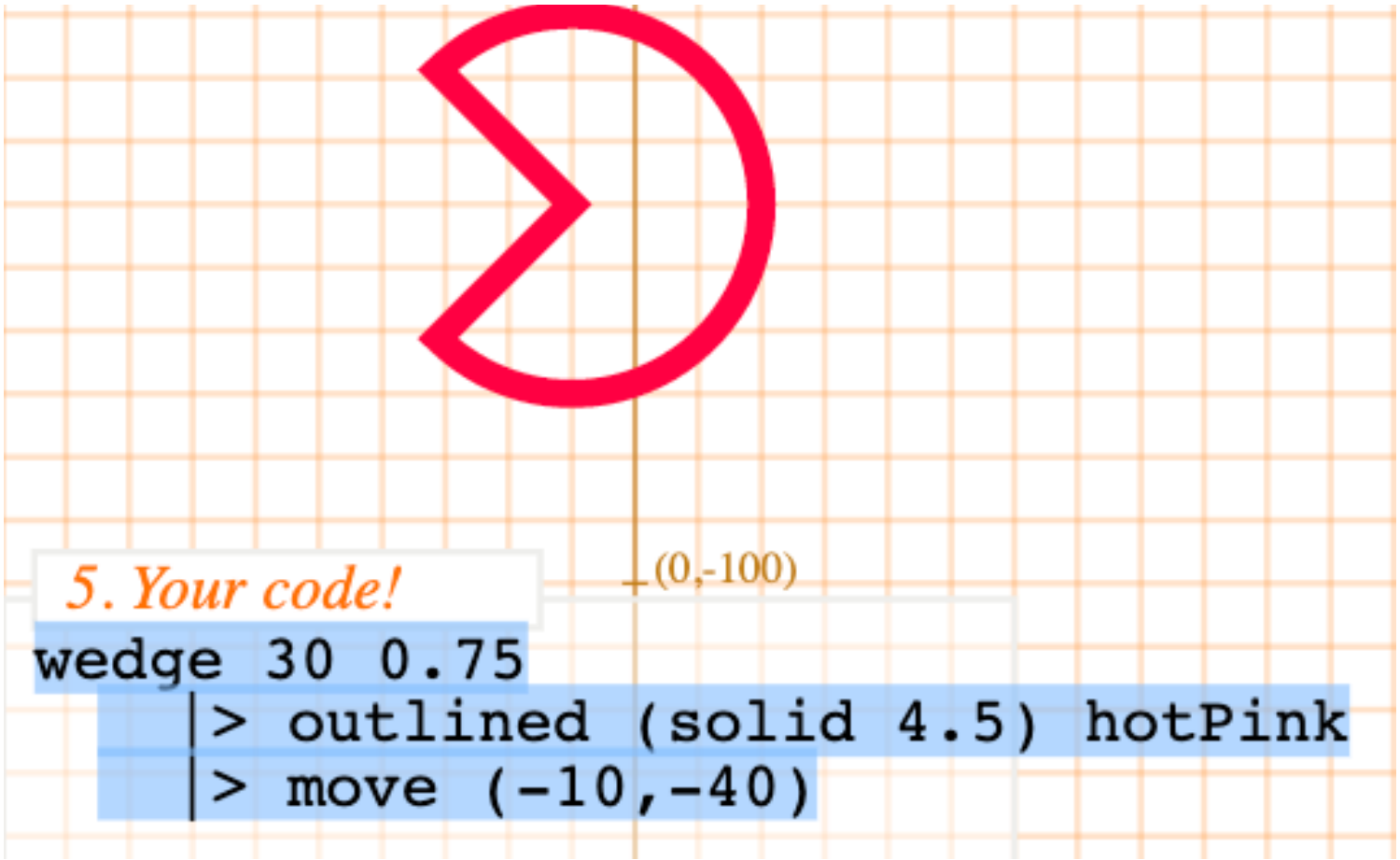
- Login
  - https://macoutreach.rocks/
- Slots for
  - Pictures
  - Games
    - Animation
    - Interaction
    - Much more

Integrated Development Environment

- Code on left
- Result on right
- Documentation links on top
  - GraphicSVG docs
  - Shape Creator live docs
  - Game Examples
  - Cool Demos

Create in Shape Creator

- Try to duplicate this
- Ctrl-A Ctrl-C, switch tabs and paste into list



Paste into IDE

- Paste into `myShapes model = [ … ]`

```
35
36
37
38
39 myShapes model =
40   [ circle 20
41       |> filled red
42   , wedge 30 0.75
43       |> outlined (solid 4
44       |> move (-10,-40)
45   ]
46
47
48
49
50
```

Make it Move

- Use model.time in place of any number
- Try changing the x-component of the move vector

```
myShapes model =
  [ circle 20
      |> filled red
  , wedge 30 0.75
      |> outlined (solid 4.5) hotPink
      |> move (-10 * sin model.time,-40)
  ]
```

Group It

- `group` makes a [list of shapes] into a compound shape
- Now you can move or rotate it together

```
myShapes model =
  [ group
      [ circle 5
          |> filled red
          |> move (7,12)
      , wedge 30 0.75
          |> outlined (solid 4.5) hotPink
      ]
      |> rotate (degrees 30)
  ]
```

Make a Definition

```
player =
  group
    [ circle 5
        |> filled red
        |> move (7,12)
    , wedge 30 0.75
        |> outlined (solid 4.5) hotPink
    ]

myShapes model =
  [ player
      |> rotate (degrees 30)
  , player
      |> move (30,30)
  , player
      |> rotate (degrees 60)
      |> move (-30,-30)
  ]
```


```
player =
  group
    [ circle 5
        |> filled red
        |> move (7,12)
    , wedge 30 0.75
        |> outlined (solid 4.5) hotPink
    ]

myShapes model =
  [ player
      |> rotate (degrees 30)
  , player
      |> move (30,30)
  , player
      |> rotate (degrees 60)
      |> move (-30,-30)
  ]
```

Make a Function

```
player t =
  group
    [ circle 5
        |> filled red
        |> move (7,12)
    , wedge 30 ( 0.75 + 0.1 * sin t )
        |> outlined (solid 4.5) hotPink
    ]

myShapes model =
  [ player model.time
      |> rotate (degrees 30)
  , player model.time
      |> move (30,30)
  , player model.time
      |> rotate (degrees 60)
      |> move (-30,-30)
  ]
```


```
player t =
  group
    [ circle 5
        |> filled red
        |> move (7,12)
    , wedge 30 ( 0.75 + 0.1 * sin t )
        |> outlined (solid 4.5) hotPink
    ]

myShapes model =
  [ player model.time
      |> rotate (degrees 30)
  , player model.time
      |> move (30,30)
  , player model.time
      |> rotate (degrees 60)
      |> move (-30,-30)
  ]
```

if-then-else

```
player t =
  group
    [ circle 5
        |> filled red
        |> move (7,12)
    , wedge 30 ( 0.75 + 0.1 * sin t )
        |> outlined (solid 4.5) hotPink
    ]

myShapes model =
  [ player model.time
      |> rotate (degrees 30)
  , player model.time
      |> move (30,30)
  , player model.time
      |> rotate (degrees 60)
      |> move (-30,-30)
  ]
```


```
player t =
  group
    [ circle 5
        |> filled red
        |> move (7,12)
    , wedge 30 ( 0.75 + 0.1 * sin t )
        |> outlined (solid 4.5) hotPink
    ]

myShapes model =
  [ player model.time
      |> rotate (degrees 30)
  , player model.time
      |> move (30,30)
  , player model.time
      |> rotate (degrees 60)
      |> move (-30,-30)
  ]
```

if-then-else

- If-then-else produces a value
- Any value!

```
circle 5
  |> filled ( if sin t < 0 then
                darkGreen
              else
                darkRed
            )
circle 5
  |> ( if sin t < 0 then
         outlined (solid 2) darkRed
       else
         filled darkRed
     )
```

Make a Recursive Function

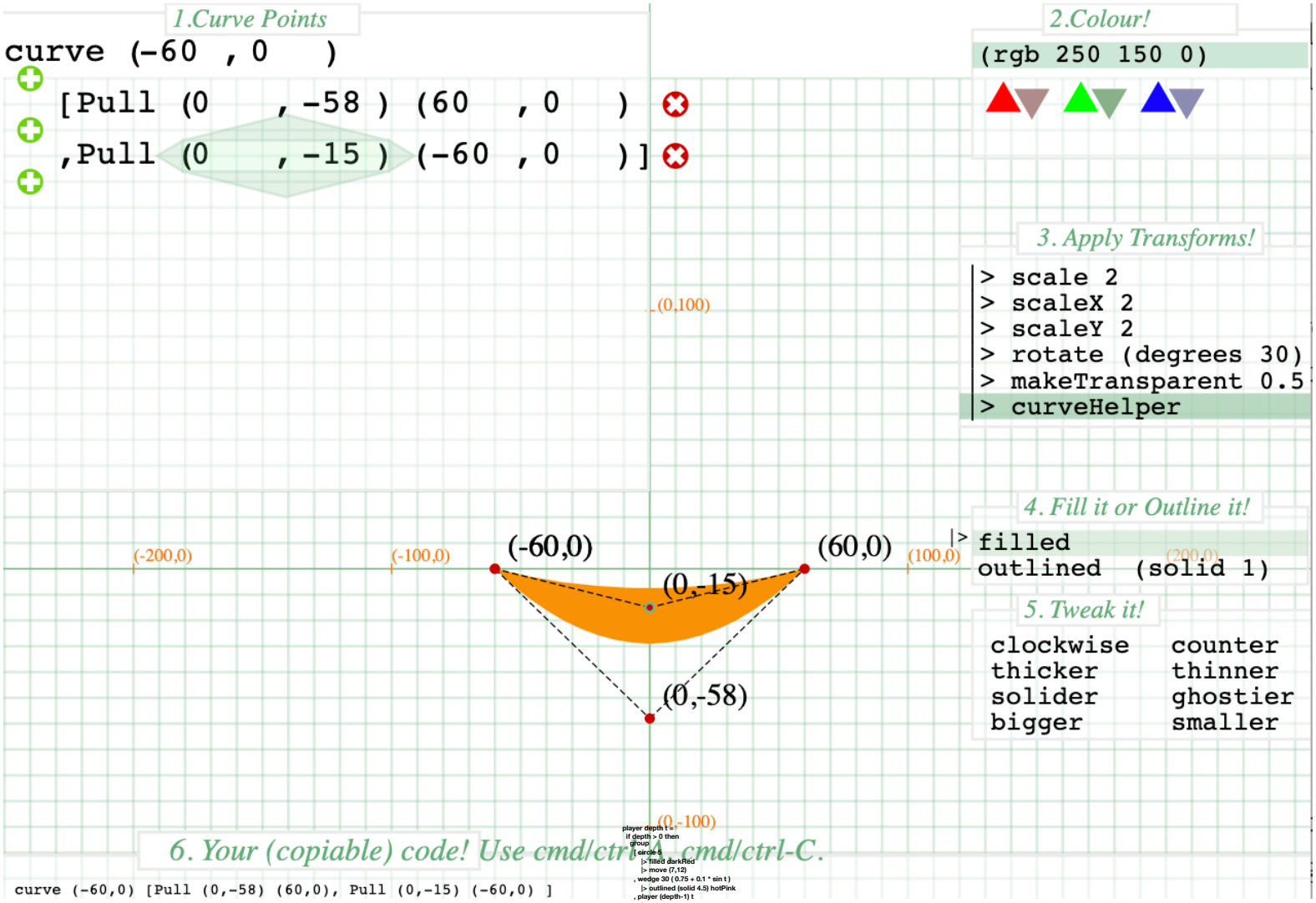
```
player depth t =
  if depth > 0 then
    group
      [ circle 5
          |> filled darkRed
          |> move (7,12)
      , wedge 30 ( 0.75 + 0.1 * sin t )
          |> outlined (solid 4.5) hotPink
      , player (depth-1) t
          |> scale 0.5
          |> move (40,0)
          |> rotate (degrees 30)
      , player (depth-1) t
          |> scale 0.5
          |> move (35,0)
          |> rotate (degrees -30)
      ]
  else
    group []

myShapes model =
  [ player 3 model.time
      |> rotate (degrees 30)
  ]
```

player depth t =
 if depth > 0 then
  group
   [circle 5
      |> filled darkRed
      |> move (7,12)
   , wedge 30 (0.75 + 0.1 * sin t)
      |> outlined (solid 4.5) hotPink
   , player (depth-1) t
      |> scale 0.5
      |> move (40,0)
      |> rotate (degrees 30)
   , player (depth-1) t
      |> scale 0.5
      |> move (35,0)
      |> rotate (degrees -30)
   ]
  else
   group []

myShapes model =
 [player 3 model.time
    |> rotate (degrees 30)
 ]

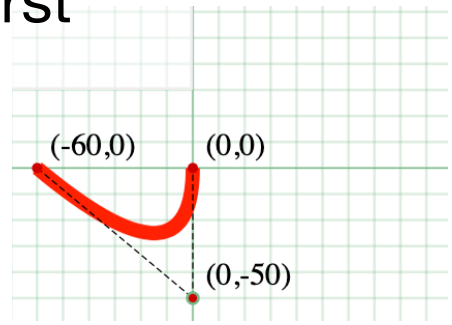
Make Curve


1.Curve Points
curve (-60 , 0)
[Pull (0 , -58) (60 , 0)
,Pull (0 , -15) (-60 , 0)]
2.Colour!
(rgb 250 150 0)
3. Apply Transforms!
> scale 2
> scaleX 2
> scaleY 2
> rotate (degrees 30)
> makeTransparent 0.5
> curveHelper
4. Fill it or Outline it!
|> filled
outlined (solid 1)
5. Tweak it!
clockwise
counter
thicker
thinner
solider
ghostier
bigger
smaller
(0,100)
(-200,0)
(-100,0)
(-60,0)
(60,0)
(100,0)
(200,0)
(0,-15)
(0,-58)
(0,-100)
6. Your (copiable) code! Use cmd/ctrl-A, cmd/ctrl-C.
curve (-60,0) [Pull (0,-58) (60,0), Pull (0,-15) (-60,0)]
player depth t =
if depth > 0 then
group
[circle 5
|> filled darkRed
|> move (7,12)
, wedge 30 (0.75 + 0.1 * sin t)
|> outlined (solid 4.5) hotPink
, player (depth-1) t
|> scale 0.5
|> move (40,0)
|> rotate (degrees 30)
, player (depth-1) t
|> scale 0.5
|> move (35,0)
|> rotate (degrees -30)
]
else
group []
myShapes model =
[player 3 model.time
|> rotate (degrees 30)
]

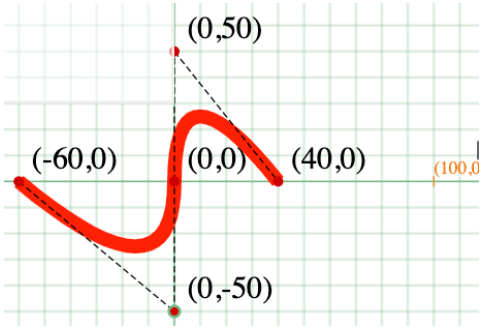
Pulls

- From first point to last, pulled towards middle point
- Filling is best if the last point matches the first

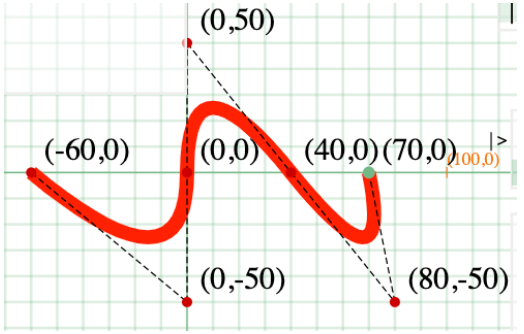
```
curve (-60 , 0   )
  [Pull (0    , -50 ) (0    , 0   )]
```




```
curve (-60 , 0   )
  [Pull (0    , -50 ) (0    , 0   )
  ,Pull (0    , 50  ) (40   , 0   )]
```




```
curve (-60 , 0   )
  [Pull (0    , -50 ) (0    , 0   )
  ,Pull (0    , 50  ) (40   , 0   )
  ,Pull (80   , -50 ) (70   , 0   )]
```



Copy and Paste

- Ctrl-A, Ctrl-C will copy the code
- Paste it into any list of shapes
- Edit numbers as needed
- Make numbers time-dependent

```
myShapes model =
  [ curve (-60,0) [ Pull (0,-58) (60,0)
                  , Pull (0,-15) (-60,0) ]
      |> outlined (solid 5.5)
                  (rgb 255 32 0)
  ]
```

Challenge 1

- In any Game Slot, make a growing plant
- Criteria
  - Be visually interesting
    - Use colour
    - Use multiple shapes
    - Use curves
  - Use good programming techniques, for example
    - use a recursive function to draw a tree
    - Use definitions
- Submit by texting “Please submit to the Hall of Fame”
  - When asked for the title, reply “ChallengeOne”

Challenge 2

- This afternoon, we will learn about interaction
- In a different Game Slot, you will make an interactive scene
- By clicking or using the keyboard, this will happen, like
  - Plants will grow
  - Flowers will bloom
  - Leaves will fall
  - Wind will blow petals
- Submit by texting “Please submit to the Hall of Fame”
  - When asked for the title, reply “ChallengeTwo”