Classwork: CompSci 94
Getting Started with Assign 6

What properties to use?

What procedures to write?

What functions to write?
Assignment 6 - Properties

- **Scene Properties**
  - Array of ancientTemplePillars
  - Array of bells
  - Array of skulls
    - VERY Important – make sure items are in order. The second item in each array is everything associated with the second pole: bell2, skull2, ancientTemplePillar2
  - Array of TextStrings (for the random code)
  - textString for game control

---

THINK EVERYWHERE about how to USE your arrays

- Can you use a loop to iterate through your array?
- Which loop?
  - For each in
  - Each in together
  - Loop with indexing
- You should NEVER rebuild your arrays.

---

Counter is as before

- TextString to display counter
- Number variable to store counter number value
- Procedures
  - setupCounter
  - updateCounter

---

SCENE Procedures

- **GenerateRandomCode**
  - Randomly assign values for your textString array that stores your code.
  - Just generate one random code, don’t worry about whether or not it is “DownDownDown” here. We will check that later
  - Use an array loop to build the code
  - Don’t use blanks! It will make it easier.
  - Capitalize words or not is fine. Be consistant
Scene Procedures (2)

- **updatePole**
  - Pass in the index of one pole and move the bell on that pole
  - You will just an if statement, but will need to index into arrays
  - Note you are only moving one bell

Scene Procedures (3)

- **Ending**
  - Game is over, determine if they won or lost and call winGame or loseGame

- **WinGame**
  - What to do if they won the game

- **LoseGame**
  - What to do if they lose the game

Scene Function - getCode

- **getCode** – returns the code as a string
  - Should build the string by walking through the code array.
  - OK to have NO BLANKS IN THE STRING
  - Example: returns “UpDownUp”

Scene Function - isGameOver

- **isGameOver** returns True if game is over, else returns false
- Determines if the current position of the bells on the poles match the code
- Use Loop with index to build a string that represents the position of the bells
- Compare this string to code string to see if they are the same
Game Control States

• “pregame”
  – Do all the setup, generate random code, give instructions
• “setup”
  – Now they can click on the tortoise only
• “play”
  – Now they can click on the poles or bells and play
• “ending”
• Game over, nothing clickable

Events

• sceneActivated
  – This will run in parallel with myFirstMethod’
  – Generate a random code by calling your procedure
  – As long as the code is “DownDownDown“ then generate a different code

Events (2)

• Event for clicking on a pole
  – This event should only happen when playing the game
  – Determine which pole you clicked on (use array index loop), and call updatePole for that pole
  – Update score
  – Determine if game is over and if so call ending

Other events

• Event for bells, similar to event for poles
• Event to show the code
• Event to click on tortoise
  – Start the game
Putting it all together

• Setup the scene
• Create all the Scene properties
• Setup your counter
• myFirstMethod
  – Welcome to the game, give instructions (If they want them), then tell them to click on tortoise to start (and change the state).
  – Don’t need anything else in here

Putting it all together (2)

• Write procedure to generate random number
• Create event to call generate random number – making sure it is not down down down
• Create secret event to show code
• Create event to click on tortoise and the game starts!

Putting it together (3)

• Create your winGame, LoseGame and Ending procedures
• Create your event to click on poles. Make sure it works
• Then create event to click on bells.
• You are done!