**WacPac Tutorial**

WacPac is a game in which the player tries to “wack” pacmen by clicking on them before they can escape to the edge of the game level. The game keeps track of kills (number of pacmen wacked) and escapes (number of pacmen that got away). The player’s goal is to keep the number of kills higher than the number of escapes. Kills become gradually more difficult over time because the pacmen move faster and faster. When escapes is greater than kills, the game is over. Your final value of kills is your score for the game.

1. Log into The Game Commons at http://thegamecommons.org. Click on the Share menu item. You will see a listing of game made in Galatea. Scroll through the games until you come to the game “Maze of Magic and Death”. Click on the Share link on the right hand side of the row entry for this game. Now Galatea will launch. Galatea will be ready to use when you see a popup stating “Galatea Initialization Complete”. At this point, you should see MazeGame in the Workspace area at the upper left side of Galatea. You will reuse the sprites and sounds from the MazeGame for the WacPac game.

2. In Galatea, use the Game-New menu item to create a new game. Go to the Game Properties pane and change the Name of the game to WacPac. Set the width and height to 768 and 624 respectively. Set the background color to any color but light gray. Rename the first level to PlayLevel and the second level to GameOver. Delete the third level.

3. Activate the Documents work area. Change the title of the game to WacPac. Copy the introductory paragraph above into the Game Description area. Put in something suitable for the Player Instructions.

4. Using the Library menu, select Temporary Sprites. Drag the following five sprites into the Sprite Palette: Brick, Block, Port, Pacman and StartButton. Now use the Library menu to select Temporary Sounds. Press Import to bring the Whistle, Pop and Buzz sounds to the Sound Palette.

5. Activate the Layout work area. In the Interface Settings pane, set Grid Type to Rectangular and set Snap To Grid to Yes. Drag the Port sprite from the Sprite Palette to the center of the canvas. Activate the Data work area. Create a new variable called “center” and set its data type to Port. Activate the Layout work area and Ctrl-Click on the Port. In the Sprite Instance Properties pane, set the instance name of the Port to center. Now drag a Brick from the Sprite Palette to the center of the canvas, placing it over the Port at the center. Use the scale tool to enlarge the Brick until it measures six grid units on a side. Use the Data work area and the Sprite Instance Properties pane to store the Brick in a variable of type Brick called “screen”. With the screen selected in the Layout work area, use the half-occluded star tool to put the screen behind the center (Port) sprite. Now you can see both the center and the screen.

6. Activate the Controls work area. In the Controllers pane, create a new Initialization controller and set its Sprite Class to Port. In the Actions pane, create a new SetVisibility action; set its Sprite Instance to center; and set Visible? to No. The port will not be visible during the game. Next make a Signal action and enter GeneratePacman as the Name of the signal.
7. Using the Controllers pane, create a new Signal controller and enter GeneratePacman for the Name of the signal that this controller will handle. Using the Actions pane, make a new CreateSprite action and set the Sprite Class to Pacman. Use the Data work area to define a variable called “pacman” of type Pacman. Return to the Controls work area and set the variable of the CreateSprite action to pacman. The new Pacman will be stored in this variable. Next make a MapToLandmark action and set Sprite Instance to pacman and Landmark Instance to center. Leave the other properties unchanged. This action will place the pacman at the center of the game level. Now make a new SetDepth action; set its Sprite Instance to pacman; set Depth Marker to screen; and set Relative Depth to Under. When the pacman is generated it will be placed under the screen so that the player cannot see it right away.

8. In the Data work area, define new level variables called “direction” and “speed”, each with type Integer, to store the velocity of the pacman in polar coordinates. Set speed to 100 and leave direction unchanged. In the Controls work area, in the Actions pane, add a RandomizeInteger action setting the Variable to direction and the Lower/Upper bounds to 0 and 360 respectively. This will generate a random direction between 0 and 359 degrees and store it in the direction variable. Now add a SetVelocityPolar action with Sprite Instance set to pacman; Speed set to Reference/speed and Heading set to Reference/direction. Now define a level variable called “speedIncrement” of type Integer with a value of 10 or so. Add an IncrementValue action with Variable set to speed and Value set to Reference/speedIncrement. This action will add speedIncrement to speed so that the next pacman will be faster. Make an AlignRotation action with Sprite Instance set to pacman. This will rotate the pacman so it faces its direction of motion. Make a GoToFrame action with Sprite Instance set to pacman; Frame Name set to startOpen and Animation set to Play. The pacman will commence animating from its startOpen frame. Finally, make a SetTimer action; set Delay to Reference/delay where delay is a new level variable of type Number with a value of 5.0 or so. Enter GeneratePacman for the Signal name. When the timer goes off, a new Pacman will be generated.

9. In the Controllers pane, make a new MouseClick controller with the Sprite Class set to Pacman. In the Actions pane, add a PlaySound action to play the Pop sound, with zero repeats. Add a RemoveFromLevel action with Sprite Instance set to clickedSprite. This action will remove the clicked sprite from the game level.

10. Use the Play menu in the Game area to play the Current Version of your game. If the game is generated successfully, you should see Pacman objects periodically emerge from behind the screen and head for the border. If you click them they will vanish.

11. Activate the Layout pane and the Shared Sprites library. Drag the Block sprite to the canvas. Scale the Block in the horizontal direction to be as wide as the entire game area. Place the scaled block at the top of the game area. Click on the Clipboard tool to copy the Block. Now click again on the canvas to place a copy of the Block on the canvas. Use the Move tool to place the new block near the bottom of the game area, leaving the last two rows of grid tiles clear. These two rows will be used to display the score. Drag another Block from the Sprite Palette to the canvas. Use the Scale and Move tools to position two copies of this brick to close off the sides of the game area.
12. Use the Text tool to create four text fields containing the following text, without the quotes: “Kills:”, “0”, “Escapes:” and “0”. Use the Data work area to define two variables called “killsText” and “escapesText”, each with data type of TextField. Use the Sprite Instance Properties pane to store the text fields holding “0” (the initial values of kills and escapes) in the two text field variables. Use the move tool to arrange the labels and zero-valued text fields into position along the bottom of the game area.

13. Use the Data work area to create a new variable called “kills” of type Integer with an initial value of zero. Return to the Actions associated with the MouseClick controller that handles clicks on the Pacman objects. Add a new IncrementValue action with Variable set to kills and value set to Literal/1. The variable kills will be incremented by one each time a Pacman is clicked by the player. Add a DisplayValue action with Text Field set to killsText and Value set to Reference/kills. The new number of kills will be displayed to the player.

14. Use the Data work area to create a new variable called “escapes” of type Integer with an initial value of zero. In the Controllers pane, add a new Collision controller with 1st Colliding Class set to Pacman and 2nd Colliding Class set to Block. Add the following actions to this controller: A PlaySound action that plays the Whistle sound; A RemoveFromLevel action that removes the first colliding sprite (the Pacman) from the level; An IncrementValue action that increases the escapes level variable by one; A DisplayValue that displays the new number of escapes in the escapesText text field. When a Pacman gets away, it whistles at the player tauntingly and the number of escapes increases by one.

15. Create a new GameCondition controller that executes when the variable escapes is greater than the variable kills, indicating that the game is over. The controller should have one action. This action will set a timer to generate a GameOver signal after a one-second delay. Create a new Signal controller to handle the GameOver signal. It should have two actions. The first action Plays the Buzz sound with two repeats. The second action uses GoToLevel to take the game to the GameOver level. Put whatever you like in the GameOver level.

16. Now test your game. If it is generated successfully, see how many kills you get before the game is over. Try adjusting speedIncrement and delay to make the game easier or harder.

17. Finally modify the game to include a StartButton object in at the bottom center of the game area. Create a MouseClick controller for the StartButton class. Add an action invoking the GeneratePacman signal to this controller, and remove the existing signal that does this in the Port Initialization controller. Now the game won’t start until the user presses the start button.