**1) Explanation of how to get your code to run**

Execute the only cell in the Project.nb file. A new full screen notebook window should be created that shows the Pacman game graphic. If Mathematica encounters an error, re-execute the cell. Pacman can be controlled by the arrow keys. Press Alt-F4 on Windows or Ctrl-W on Mac to end the game window.

**2) Proposal**

See attached document for the printed report, and see ProjectProposal.nb for the electronic copy.

**3) Schedule of Future Work**

April 27:

Collisions and multiple speeds are implemented. The speed of each entity will depend on their current mode and the level the game is currently on.

April 29:

Multiple maps: Several different map layouts .txt files, as well as possibly randomly generated maps. The main menu is changed to be more aesthetically pleasing.

May 4:

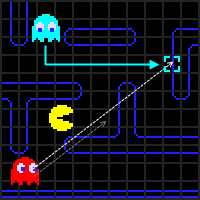
Graphics:

An AI mode for Pac-Man is implemented, in which Pac-Man moves without the guidance of a human player. This may use Minimax.

**4) Algorithms/Formulas**

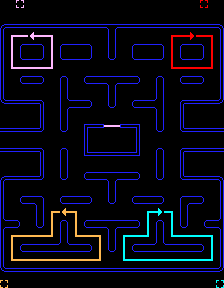
One algorithm we have learned about and applied is AI for the ghosts, as present in the game of Pacman. Each ghost has 3 modes: Chase, Scatter, and Frightened.

In chase mode, each ghost has a target tile they are attempting to reach. This target tile is different for each ghost. In the above code, we calculate the target tiles for a ghost, depending on which ghost the function is called with. Should we call getTargetTile[1], we call for Blinky’s target tile, which is simply Pacman’s location. getTargetTile[2] returns Inky’s target tile, which is the position at twice the vector from Blinky to two tiles ahead of Pacman, as shown in the image.



Ghost 3, Pinky, aims to reach 4 tiles ahead of Pacman. Ghost 4, Clyde, attempts to reach Pacman if his distance from Pacman more than 8 tiles away, but aims for his “ghostCorner” tile if he is less than or equal to 8 tiles away from Pacman.

This defines each ghost’s corner is defined based on the image below. These tiles are as the ghosts’ target tiles while when the ghosts are in Scatter mode to disperse them away from Pacman. This was used to release the pressure off the player and make the game more enjoyable.



The final mode, Frightened mode, is initiated when the player eats a power dot. When this occurs, we reverse the ghosts’ directions and start a scheduled task that turns off Frightened mode in 5 seconds. During Frightened mode, each ghost does not have a target tile, and instead makes random decisions for each direction change.

**5) Files and file structures**

map3.txt contains the map data, with the following associations between characters and meanings:

# is a wall

X is a power dot

. is a normal dot

P is Pac-Man’s starting position

i is a special intersection point that will later be implemented for more accurate ghost movement. Its effects are not very noticeable to the average player.

b is a black tile that cannot be walked upon

G is a ghost spawn tile

\_ is a path tile that does not contain a dot

w is a warp tile

“pacman - Copy.png” contains the Pac-Man sprite used in-game

“Pacman-death-sound.mp3” is the sound played upon Pac-Man’s death

“Start Music.wav” is the start music played upon game start-up

**6) Data structures**

path, warps, specialIntersections, and ghostSpawns are constant lists that represent data read from the map

dots, power, pacman, and ghosts are lists whose initial contents are read from the map, but whose values change as the game progresses. I.e. a dot’s coordinates will be removed from the dot list when Pac-Man “eats” the dot

List ghostsDir represent the current movement directions of the ghosts

List isFrightened represents whether each ghost is frightened

Boolean shouldScatter represents whether the ghosts should scatter, determined by the checkbox on the main menu

ghostColors represent the colors of each ghost, in order

List dir represents the current movement direction of Pac-Man

List nextDir represents the next movement direction of Pac-Man

score stores the current score, which increases when dots or ghosts are eaten.

level stores the number of the current level

score stores the value of the current game score.

scatterTimes stores the seconds that will elapse before the ghosts’ movement mode changes to scatter. This is determined by which level the game is currently on, so there are three element lists in scatterTimes: one for level 1, one for levels 2-4, and one for levels 5+. Each of these sublists contain integer elements that represent the seconds that must elapse before the mode toggles from scatter to chase or vice versa.

directions stores the coordinate additions for each of the possible movement directions.

actions is a list of associations from events such as key presses to functions such as updateMovement[], to be used in the EventHandler

intersections stores a list of the points on the path at which there are more than two possible directions to turn.

deathCounter stores the number of times pacman has died.

killCounter stores the number of ghosts pacman has killed each time frightened mode is active, allowing an increase in score for each subsequent kill.

**7) Code approaches to problem**

One aspect we have learned about and applied is AI for the ghosts, as present in the game of Pacman. Each ghost has 3 modes: Chase, Scatter, and Frightened.

updateMovement[movement\_] :=

If[MemberQ[path, pacman + movement], dir = movement,

nextDir = movement];

This method is called when an arrow key is pressed to move the Pac-Man character, and changed Pac-Man’s current direction and his next direction.

getTargetTile[i\_] := If[isScattering, ghostCorners[[i]],

Switch[i,

(\*blinky always goes for pacman\*)

1, pacman,

(\*go for the tile that is at blinky plus 2 time the distance \

between 2 spots ahead of pacman and blinky\*)

2, ghosts[[1]] + 2 (pacman + 2 dir - ghosts[[1]]),

(\*go for 4 tiles ahead of pacman\*)

3, pacman + 4 dir,

(\*if the distance between it and pacman is greater than 8,

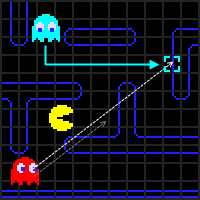
go for pacman. else, go for the bottom left corner\*)

4, If[EuclideanDistance[ghosts[[i]], pacman] > 8, pacman,

ghostCorners[[4]]]

]

];

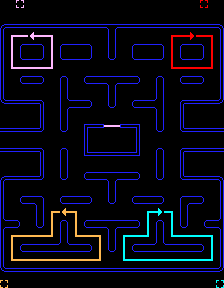
In chase mode, each ghost has a target tile they are attempting to reach. This target tile is different for each ghost. In the above code, we calculate the target tiles for a ghost, depending on which ghost the function is called with. Should we call getTargetTile[1], we call for Blinky’s target tile, which is simply Pacman’s location. getTargetTile[2] returns Inky’s target tile, which is the position at twice the vector from Blinky to two tiles ahead of Pacman, as shown in the image.Ghost 3, Pinky, aims to reach 4 tiles ahead of Pacman. Ghost 4, Clyde, attempts to reach Pacman if his distance from Pacman more than 8 tiles away, but aims for his “ghostCorner” tile if he is less than or equal to 8 tiles away from Pacman.

ghostCorners = {{2,

Length@map[[1]] + 2}, {Length@map - 1, -2}, {Length@map - 3,

Length@map[[1]] + 2}, {0, -2}};

This defines each ghost’s corner based on the image below. These tiles are as the ghosts’ target tiles while when the ghosts are in Scatter mode to disperse them away from Pacman. This was used to release the pressure off the player and make the game more enjoyable.



The final mode, Frightened mode, is initiated when the player eats a power dot.

ghostsDir = -ghostsDir;

Upon eating the power dot, the ghosts’ directions are reversed.

turnOffFrightened = CreateScheduledTask[

isFrightened = Table[False, {4}],

{5}

];

When this occurs, we also start the scheduled task shown above, which turns off Frightened mode in 5 seconds. During Frightened mode, each ghost does not have a target tile, and instead makes random decisions for each direction change.

The update method contains the code that is executed for each update of the game logic, scheduled by a ScheduledTask. First, we update a counter that contains the number of updates since the start of the program. Next, if scattering is turned on and it is time to scatter (as determined by the counter value relative to the times the scattering should switch), the isScattering boolean is toggled. Next, between Pac-Man and the Ghosts are checked for. If the ghosts are not frightened, then a death sound is played and the ghosts’ and Pac-Man’s positions are reset. Next, tile collisions with Pac-Man is checked; normal dots increase the score by 10 and power dots increase it by 50 and toggle frightened mode.

(\*pacman movement\*)

Which[

(\*pacman is on a warp time\*)

MemberQ[warps, pacman],

pacman = warps[[1 + Boole[pacman == warps[[1]]]]] + dir,

(\*if pacman's next direction is not blank and is on the path,

make pacman go in the next direction and the current direction \

become the old next direction. the next direction is now blank\*)

nextDir != {0, 0} && MemberQ[path, pacman + nextDir],

pacman += nextDir; dir = nextDir; nextDir = {0, 0} ,

MemberQ[path, pacman + dir], pacman += dir

];

The update loop also updates Pac-Man’s movement. If Pac-Man is on a warp tile, he is moved to the other warp tile. If he is currently moving and his next location (in the same direction) would be on the path, his position is moved to the next tile.

The following code segments were modified from <http://mathematica.stackexchange.com/questions/16913/how-to-get-full-screen-graphics-canvas>:

“ImageSize -> AbsoluteCurrentValue[EvaluationNotebook[], WindowSize]” is used inside the Graphics display to display the Graphics in the largest possible size.

“WindowFrameElements -> {"CloseBox", "ZoomBox", "MinimizeBox"}, WindowTitle -> "Pac-Man", WindowSize -> Full];“ is used in the CreateDocument element to create the document with the proper frame elements, title, and window size.

**8) Description of accomplishments**

Up to the first progress report, we had implemented a map, player-controlled Pac-Man movement, ghost AI similar to actual game A, including chase and scatter modes, score based on dots eaten by Pac-Man, a frightened mode for the ghosts that changes their color and AI to make random decisions, warping when the ghosts or Pac-Man touches a warp tile, and partially implemented collisions.

Since the first progress report, we have changed each tile to be an 8x8 region, as in the actual game, which will allow for different movement speeds between Pac-Man and the ghosts. This was not a simple framework change and required adjustments to ghost AI algorithms and graphics rendering. Lives have also been implemented. Although some work was done on collisions, the new 8x8 grid system ruins this progress, though it will make collisions much better when they are implemented successfully. A main menu has been added that will provide pre game options such as whether scatter mode will occur. This menu does not look very good now, but its aesthetics will be improved soon.

Lives are implemented with a counter in the top right corner. Upon reaches a life count of 0, the game pauses indefinitely and must be stopped manually. Pac-Man’s sprite now has the odd behavior in changing to size 1 when on a dot. We are currently unsure why this occurs because Pac-Man’s inset element clearly defines his size to be 8.

**9) Signed Statements**

**Aaron Kirtland:**

Up to this point, I have accomplished a large portion of the translation of Pacman’s mechanics as described in the Pac-Man Dossier and “Understanding Pac-Man Ghost Behavior”. Specifically, I have created the framework for reading a map file into *Mathematica* and rendering dynamic graphics based on this input file. I have coded ghost AI based on the “Understanding Pac-Man Ghost Behavior” website, and have used this AI along with pathfinding algorithms to make ghost and Pac-man movement as in the original game. I have experimented with adding sound effects to the game, and have found successful startup music and computer-generated music notes for eating dots.

Since the last Progress Report, I have made an introductory main menu screen with an option to stop scatter mode from occurring, thereby making the game more difficult, and I have changed the grid size such that each square is not an 8x8 region. This major framework change will allow for the game to function much more realistically, with speed differences between the ghosts are Pac-Man dependent on the current level.

In the future, I will begin to add less essential components to the game, now that the main framework is completed, including speed differences and graphics more closely resembling the actual game. I will also research implementing AI algorithms for Pac-Man, such as minimax. These AI’s will be available for use in place of single-player based on an options menu. See the Schedule in part 2 for an approximate timeline of when these objectives will be reached.

**Caleb Stickney:**

Thus far, on this project I have taken responsibility for collisions, implemented lives, worked on sounds, added score, and researched patterns that need to be present in the game. Most of my time has been spent on collisions, something that has been significantly more difficult than expected, however this will hopefully soon change.

Since the last report I have completely implemented the score and lives mechanics as well as worked heavily on collisions.

In the future of this project I hope to continue contributing significantly. I will take continue to work on collisions, add multiple levels, and work on other non-critical aspects of the project.

**10) Progress Report 1**

See attached.

**Sources**

Pac-Man Dossier <http://www.gamasutra.com/view/feature/3938/the_pacman_dossier.php?print=1> “Understanding Pac-Man Ghost Behavior” <http://gameinternals.com/post/2072558330/understanding-pac-man-ghost-behavior>