Project 1:
Simplified Plants vs Zombies

Goals

- To make sure you can write programs using combinations of lists and structures.
- To practice using the design recipe to organize a nontrivial program.

It goes without saying (we hope), but you must use the Design Recipe for all functions. In particular,

- Don't forget to write a signature, purpose statement, and unit tests for every function.
- Write the tests before you write the function definition.
- Break up functions into smaller functions if they become too complex.
- Define every data type you use.

Plants vs. Zombies is a member of the Tower Defense genre of game: the player strategically places her defenses on the board (in this case, your plants) and these defenses protect the player from the enemy forces (in this case, zombies). Each kind of plant effects the zombies differently, but each costs energy and you only have so much to start. So choose wisely to avoid a zombie apocalypse.

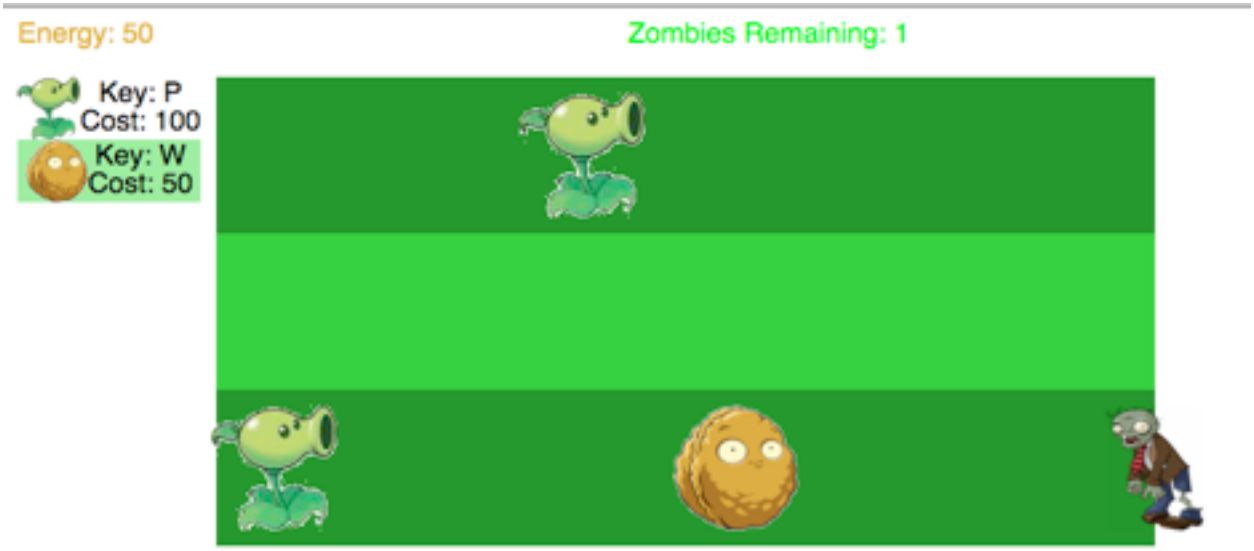
While we have dramatically simplified the game play, it still will be by far the largest program you have constructed in class. Strict adherence to the Design Recipe is required to keep everything straight. In Project 1 we still provide a large amount of structured help, but be aware than unlike a lab most design decisions will be up to you and your partner: we will not be telling you "develop this function, then that one, then...".

We clarify the minimum functionality in the game required for an "A", however we also point out many opportunities for extra credit, as this is a very extensible premise :-)

The Assignment

Your job is to create a simplified version of the Plants vs Zombies tower defense game. The player has a limited amount of solar energy, which replenishes very slowly, with which to "purchase" plants to protect themselves from the zombies. The game is played on a grass field consisting of several horizontal strips. **Zombies** enter from the right end of a strip and move left until they reach the left side and eat your brains (you lose). If a zombie encounters a plant, however, the zombie is halted while it destroys the plant. Some plants will also actively harm the zombies. There are a limited number of zombies on each strip; if all zombies are destroyed, you win the level.

In our simple version, there are two types of plants: the **WallNut**, which does nothing but sit there, but which will slow down a zombie as it will take time to destroy. The second plant is the **PeaShooter**, which, as the name implies, shoots peas left to right down the strip. A PeaShooter has a cooldown time between peas. When a **pea** hits a zombie, the zombie is injured. When the zombie's health is reduced to 0, it dies...hmm...uh, decomposes? Pressing the "P" key chooses the Peashooter, and pressing "W" chooses the Wallnut. A mouse click ("button-down") selects both a row and a specific X position (we ignore the vertical Y value, other than to select the row in which the plant is located).

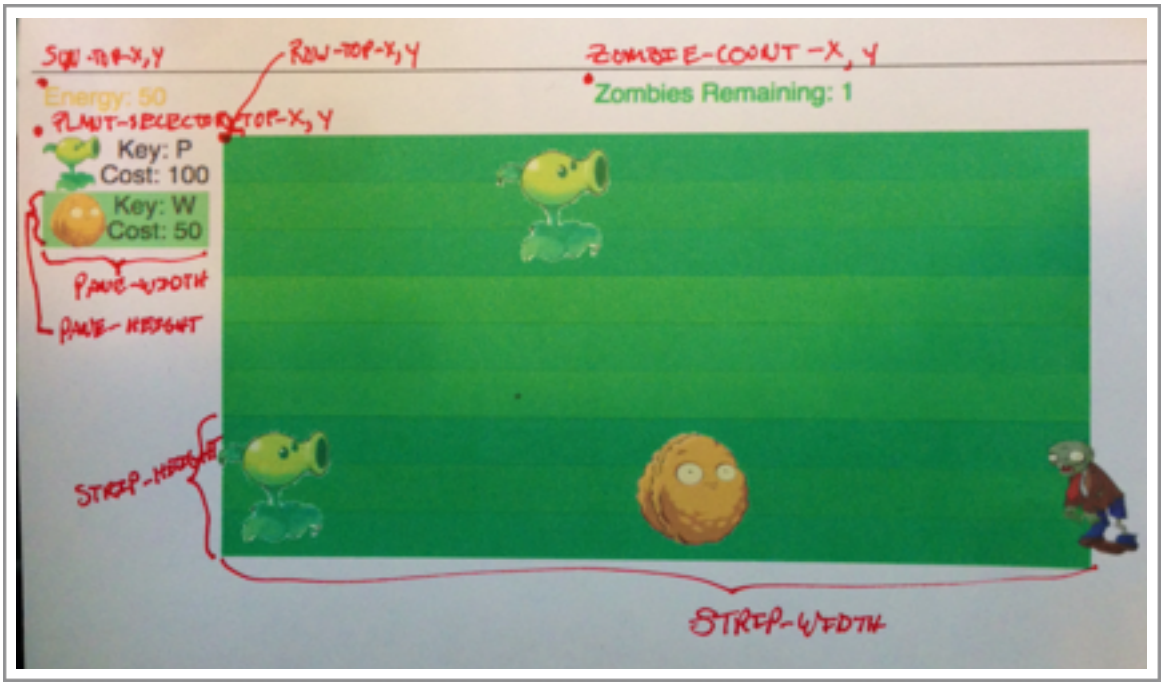


View

Below are some constants for important screen locations that map to the starter code we provide, however, feel free to design your own layout if that is of interest to you, this was thrown together very quickly. In particular the View needs to indicate to the player:

- Current solar energy levels
- Number of remaining Zombies
- What plant is currently selected for planting, and cost and key info
- Locations of all the currently active Plants and Zombies

At the end of the level, a win/lose screen should be displayed. We do not require more than a single level, or health bars on plants & or zombies, or dynamic changes in the plant selector similar to the commercial game (however, see Extra Credit Extensions). In particular, the plant selector view can simply be two static images (one with the PeaShooter selected and one with the WallNut selected).



Controllers

Keyboard

The keyboard should control plant selection, a “p” or “P” to select the Peashooter, and a “w” or “W” to select the WallNut. Other keys can be ignored (see Extra Credit Extensions).

Mouse

The `“button-down”` event should control the placement of a new plant, of the currently selected plant type (see Keyboard). However, each plant type has an energy cost, and the user must have enough energy to buy that plant. The cost of the plant must be subtracted from the user’s available energy. Also, if the user clicks somewhere that plants cannot be planted (i.e., off the grassy strips) then the mouse is ignored.

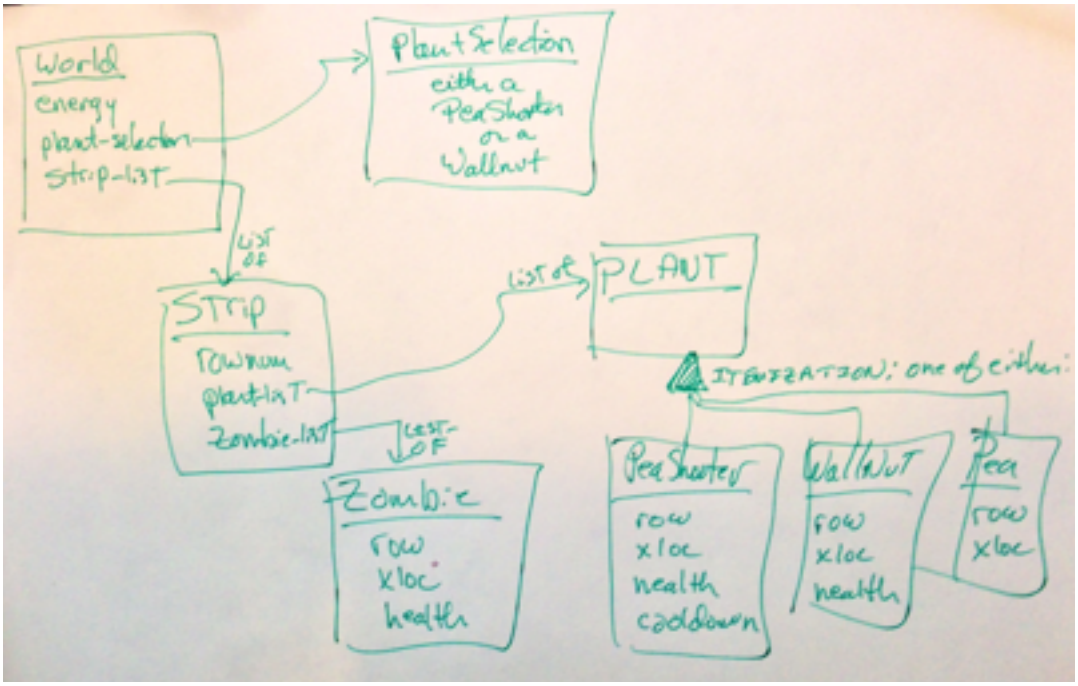
Clock Tick

Each clock tick involves several things in the **full** project:

- Replenish energy at the current energy-replenish-rate.
- Track the cooldown time for each Peashooter. If a Peashooter has reached its cooldown time, create a new Pea at that Peashooter
- Move every Pea forward. If a Pea hits a Zombie, reduce the Zombie’s health (the Pea expires).
- Move every Zombie forward, if not blocked by a plant. If a Zombie is blocked by a plant, the Zombie does damage to that plant’s health.
- Remove any decomposed Plants and Zombies.

Model

You will need to create Data Definitions for all of the elements in the game. A good way to start on this or any project is to draw out some ideas on a whiteboard and discuss (Post-It notes also work well with each Post-It representing a structure, Itemization, or other major Data Definition. For example, here was my initial sketch [There is no need to follow this design, although you are welcome to use it if you want. Changes will likely become necessary as you begin to actually build your solution]:



Level Design

We leave the specific level design up to you: what should the initial energy be, the refresh rate, the cooldown, the initial healths, number of zombies and starting locations, speed, Etc. Etc. Etc. If you aren't really interested, pick anything plausible. If you are interested in Game Design, now is the time to try your prototype on your dorm mates. What values make the game fun to play? Hard to win but not impossible?

Suggestions

Software Engineering Strategy

If you and your partner prefer to work separately, work out and agree on the data definitions first, then proceed to work on the rest of the code (together or independently). In a professional setting, one person or team is responsible for each data definition ***and all the functions that operate on that data definition***. Function signatures become literal *contracts* between you and your teammates.

Implement the game in stages ("increments"), not all at once.

- Get the basic Data Definitions down, and implement rendering functions for everything so that you can draw the initial world
  - If you want, you can support only **one** strip of grass to start.
- Implement the keyboard plant selector to choose between planting WallNuts or PeaShooters
- Implement the mouse-handler to plant a selected plant, and update the Energy display. *[Up to here is Lab 5]*
- Implement energy-replenishment.
- Implement Zombie behavior (move left until near plant, damage plant until gone, continue)
- Implement win/lose termination (zombie reaches left edge = lose, all zombies dead = win) *[All behavior up to here is a C grade]*
- Implement Peashooters, cooldown, Pea movement and attack, zombie reaping. *[Grade B]*
- *[Add in multiple strips at some point for full credit A]*

You will get more credit for a fully-working, well-designed solution to part of the game than for a non-working solution to the whole game.

Hints

You will be graded primarily on the structure of your code, not on efficiency. Don't worry about efficiency. Your computations to check when a zombie is `close?` to a plant (or pea `close?` to a zombie) do not have to be pixel-perfect. If the resulting game play looks plausible, we won't care about the precise formula you used.

What to turn in

Turn in a single file project1.rkt containing all code and documentation for this assignment. Make sure that both students' names are in a comment at the top of the file. Projects may be submitted late, at a penalty of 15% of your grade per day. (Yes, weekends count).

Extra Credit

This project lends itself to many extensions. If you care about exactly how much credit we will award, please check with us first. In general we won't award more than 50 points extra even if you re-implement the full game :-)

You can get ideas by looking at the commercial game (PVZ2 is free), or websites describing the game, plants, zombies, and game strategies for both versions:

- New types of plants with different behavior (e.g. SunFlowers cause higher energy replenish rate)
- New types of Zombies
- A more dynamic view (health indicators, plant selector visual refill cues, etc.)
- Multiple levels
- External level definitions (read in from a file)