

## **1. Project Description**

The name of my term project is “Space Typers.” It is based on children’s games that are meant to help kids learn how to type. In the game, the user will have to shoot aliens– but every time an asteroid falls, it will have a word written across it, and the user will have the option of clearing all the aliens for a short period of time if they type the word correctly. The game will also track what words each registered player tends to miss more often.

## **2. Competitive Analysis**

My project is inspired by typing games that I used to play in elementary school. This is quite similar to those games in the “asteroid” aspect, as those games also had falling asteroids, and I would have to copy the word that was falling. My project is different from those games since mine has enemies. In those games, I don’t recall there being a “game over” or a “win,” it would just keep going until all the asteroids had fallen. In my game, there will be falling aliens that you have to shoot down while keeping an eye out for the asteroids. My game also has a winning and losing– if a certain number of aliens reach the ground, it is game over. If the user reaches a certain point value or clears enough asteroids, they win.

My game will also store the data containing which words each user tends to miss most often. Every time a user is entered into the system, they will be stored as a player, and each player will have their own “problem words.” The program will be able to tell the user which words they miss so that the user can practice those words. I also plan to make the program drop words that the user gets correct less often, and the words they miss more often.

## **3. Structural Plan**

I’m going to have an object representing each component of the game (i.e. one for asteroids, one for aliens, one for the player etc.). The list of possible words will be stored in a separate text file, and the program will read the text file to extract the words. As of right now, I am planning on having all the core code in one file (framework and animation). There will be separate files for the images used in the game.

## **4. Algorithmic Plan**




I think the trickiest parts of my project will be training the algorithm for the words to drop more commonly missed words and keeping track of commonly missed words for each user. Each time a player registers their name before playing, they will be stored in a set as a class attribute for the Game mode. For each player, I am planning on having a dictionary of matched (correct) words and a dictionary of missed words and the frequency of how often they were missed/matched. To make missed words appear more often, I will probably modify my getRandomWord() method to check whether or not the generated word appears in matched or missed. If it appears in the matched dictionary more than twice, I will generate a new word. To

tell users which users they miss more often, I'm planning to pull the top 5 (or less if there aren't 5 words in the dictionary) most frequently missed words in their missed dictionary. I also have to create a text entry box using mousePressed and keyPressed, since tkinter's getUserInput pauses the game. I'm planning on using mousePressed to detect if the user has clicked inside the box and keyPressed to generate the string the user types into the box.

## 5. Timeline Plan




- Tp1: have the game fully functioning (collisions for shooting aliens, taking user input for the words on the asteroids)
- 12/2: Have the algorithm set up so that missed words show up more frequently and have the program be able to tell the user what words they miss more often
- Tp2: have the game complete with user data, word algorithm, and hopefully have smart aliens (moving in different directions) and player power-ups

**6. Version Control** - I'm using Dropbox to back up my code. Every time I successfully add a feature or make a significant change, I back up that version.


term-project.py [Version history](#)    

You can restore any version below to make it the current file. All other versions will still be saved.

Today

	<b>term-project.py</b> 3:52 PM	Edited by Kruthi Thangali Web	18.49 KB	Current version
	<b>term-project.py</b> 2:14 PM	Edited by Kruthi Thangali Web	18.5 KB	
	<b>term-project.py</b> 1:04 PM	Edited by Kruthi Thangali Web	17.55 KB	

Yesterday

	<b>term-project.py</b> 8:33 PM	Edited by Kruthi Thangali Web	14.65 KB	
---	-----------------------------------	----------------------------------	----------	--

**7. Module list** - I'm not planning on using any modules.

## TP2 Update

- I added a few power-ups to the game:
  - There are ice cubes that fall every 25 seconds, if you shoot one successfully then the aliens/asteroid will stop moving for 5 seconds and you can still shoot them

- If you type 3 words correctly in a row, you get a “super bullet” (the bullet will be 3 times as big), so you may be able to hit extra objects
- If you type 5 words in a row correctly, you get a ricochet bullet that will bounce off the side walls and any aliens it hits/nears until it reaches the top of the screen
- I added parabolic bullet shots, which can be activated by pressing shift+A
- I added a feature that makes the aliens move away (50% of the time) if a bullet nears it
- The aliens now move side to side and downwards
- I changed how the aliens function– now each alien has a word, and you have to type the word on the alien correctly to unlock it before you can shoot it
- Asteroids fall less often now, and each asteroid will have 3 words on it
  - All three words have to be typed correctly in order to clear the asteroid and all the current aliens
- I also added levels that are separated by word lengths
- Since I changed the game by making asteroids have multiple words and aliens also have words, I had to change how I handle missed/matched words
  - If a word is in either the alien word list or asteroid word list, then a function is called to handle that match
  - If a word is not in either list, then it gets counted as a missed word
    - Since there are multiple words that the user may have been attempting to type– I used an algorithm that goes through all the possible words and counts how many letters the typed word and the possibly attempted word have in common– the word with the highest number of common letters is detected as the attempted word

### **TP3 Update**

- There is a report feature at the end of the game– this tells you how many words you typed incorrectly, how many you typed correctly, and how many words you missed completely (i.e. floated off the screen). This also gives the user an opportunity to practice any missed/incorrect words
- There is a user data feature that tells the user what words they miss often and what words they get often
- I did not get to incorporate an algorithm that generates commonly missed words more often