|  |  |  |
| --- | --- | --- |
| **Advanced Graphics Programming** | | **Major Lab 07B** |
| **Finishing the Typocommand Game** | **70 through 110 Point Versions** | |
| **Assignment Purpose:**  Complete the final stages of the Typocommand game. | | |

Stage 07 of the **Typocommand** Case Study is functional, but lacking. Letters do fall. Typing the letters will destroy them with a laser. A city is destroyed if it is hit by a letter. Several features still need to be added to complete the game.

**70-Point Version**

For the 70 point version, you need to complete the following tasks which should create a minimal, yet playable **Typocommand** game.

**Background**

Create a static background full of random stars.

**Scoring**

The game needs to keep score. More points should be awarded if a letter is shot sooner rather than later

**The Laser Cannon’s Appearance**

Make the **LaserCannon** more than a simple triangle.

**Exploding Cities**

A **City** should explode when hit by a **Letter**, in the same way that a **Letter** explodes when shot by the **LaserCannon**. Hint: **City** inherits from **GameThing** so it too has a **triggerExplosion** and **exploding** method which can be redefined.

**Warping Letters**

Warping Letters will randomly change their location from one column to another after a short period of time. This is only a *horizontal* warp. The **WarpingLetter** will maintain the same **y** value.

**Morphing Letters**

Morphing Letters will randomly change their value to another letter/digit after a short period of time.

**Game Over**

When all of the cities are destroyed the game is over. The final score should be displayed on the screen.

**75 through 110-Point Versions**

There are a variety of other features that can be added to the **Typocommand** game.

Each of these will earn an additional 5 points for a maximum grade of 110.

**Background**

Animate the background. For example, the stars could slowly move to simulate a planet’s rotation.

**The Laser Cannon’s Appearance**

Animate the **LaserCannon**.

**Improving Explosions**

Come up with your own style for the explosions of letters and cities.

**Speeding Up**

At certain score benchmarks the game should speed up. Right now all **Letter** objects – and their subclasses – have a **speed** attribute which is set at **7**. Decreasing this value will speed up the game.

NOTE: As the game speeds up, the point value for the letters should increase.

**Special Keys**

<Up Arrow> - Increase Letter Speed

<Down Arrow> - Decrease Letter Speed

<Right Arrow> - Increase Letter Size

<Left Arrow> - Decrease Letter Size

<Home> - Restore Default Letter Speed and Letter Size

**Phantom Letters**

Phantom Letters are tricky because they are invisible; however, there reflection shows up on the opposite side of the screen.

To create the “reflection” of a letter, you create a **Font** object with a *negative point size*. Some adjustments will need to be made when you consider the orientation of the **x** and **y** values of a letter with a positive or a negative point size. The example below shows a capital ‘G’ with both positive and negative point size. The dot between the 2 Gs represents the **x** and **y** value of the letter. Notice that for a normal letter, **x** and **y** represent the lower-left corner of the letter. In the *reflection* of a **PhantomLetter**, it represents the top-right corner.

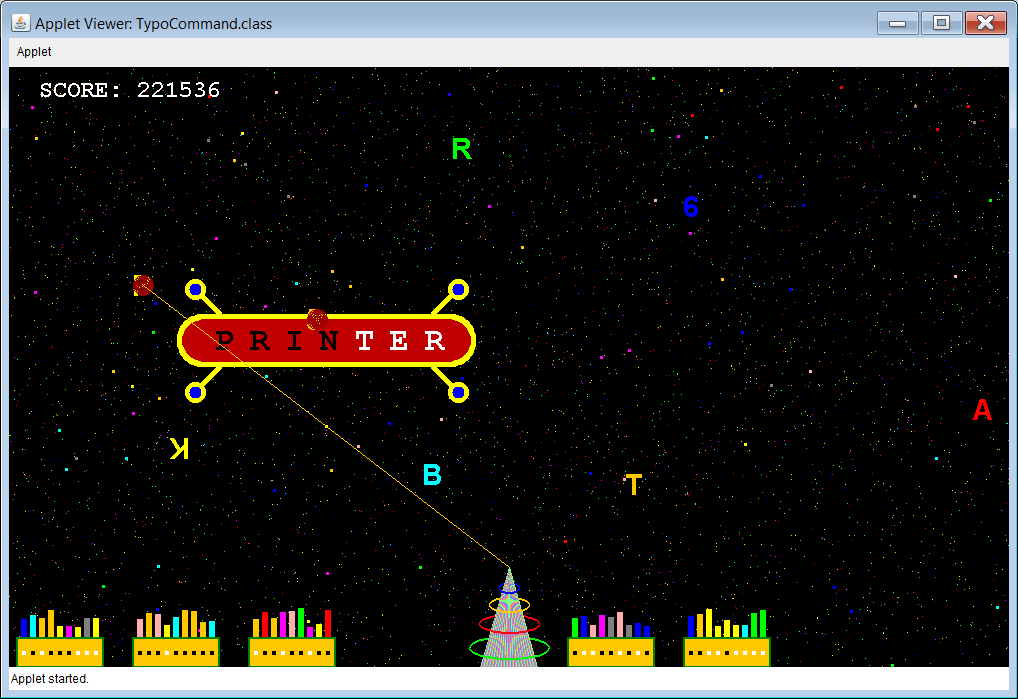
|  |  |
| --- | --- |
| With a positive point size  the letter is displayed  normally. 🡪 🡪 🡪 🡪 |  |
|  | With a negative point size  the letter is displayed  🡨 🡨 rotated 180 degrees. |

**Discourage Random Typing**

If the player types a letter that is nowhere on the screen, he/she should lose some points. If the player types 20 incorrect letters in a row, the score should be reset to 0.

**Word Blimps**

A **WordBlimp** is a **GameThing** that contains a random word and moves horizontally across the screen every few minutes. See the example below. The size of the **WordBlimp** is based on the length of the word. As each **Letter** of the *word* in the **WordBlimp** is typed, it will change color. Typing the entire word triggers an explosion for every **Letter** on or above the screen. Also, if a **Letter** crashes into a **WordBlimp**, the **Letter** explodes. A **WordBlimp**, like a **LaserCannon**is indestructible.



**City Lights**

In the image above, you will notice that some lights are on and some are lights are off. The city lights need to turn off and on at random. When a city is hit by a letter, all of its lights should go off as it is exploding.

**Sound**

Add appropriate sound effects to the game.

**High Score**

Store the names and scores of the *Top Ten Typocommanders*.

**Your Very Own Special Letter Subclass**

Create your own original subclass of Letter. For example, I created something I call a **RainbowLetter**. Follow this link and press the <Backspace> key once the game starts. You will notice some letters that behave differently than anything that has been described so far.

<http://www.schram.org/john/TypoCommand/TypoCommand.html>

Your job is not to duplicate my **RainbowLetter**. You need to come up with something completely original.

**Introducing Letter Subclasses One By One… and a Cheat Key**

At the beginning of the game, only normal **Letter** objects are falling. At certain score benchmarks different types of letters should be introduced into the game, one by one. For example, at 10,000 points introduce **FastLetter**s. At 25,000 points introduce **VanishingLetter**s as so forth. This will make sure the game is fairly easy at the beginning for novice typists.

For people who want a challenge, and do not want to wait around for the game to get challenging, add the <Backspace> cheat key. When this key is pressed, the player’s score is automatically increased by a sufficient amount so that all Letter Subclasses are in the game.

NOTE: If you have completed the **Speeding Up** feature, you will notice that this makes the game faster as well.

ALSO: If you have completed the **WordBlimp** feature, you should make the word blimps appear about 10

times more frequently.