ABLOCKALYPSE

Paulina Trifonova and Yael Borger

Guide of the Write-Up

- Lore
- Gameplay
- Controls
- Development
- Conclusion

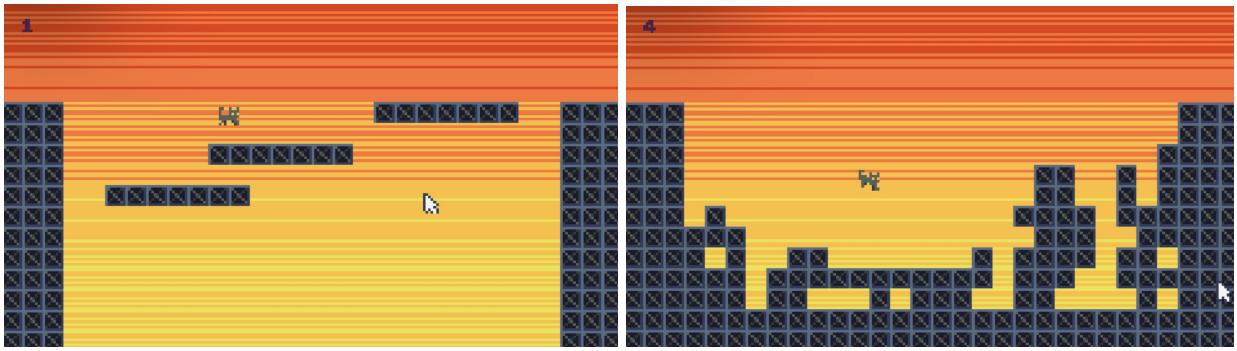
Lore

With all of the earthly items launched into the stratosphere and hunks of metal sent to scour the universe, it was beyond time for retaliation. Humans were convinced that, because Space is vast and infinite, that the trash sent would not need to be cleaned properly. Why waste resources cleaning up the mess they made when they could wait and see if their mistakes would eventually come barreling back to Earth? And that's exactly what happened. The buildings crumbled to the pressure, the sky warmed to a hellish hue, the destruction reached all parts of the planet, and now our story begins.

An unspecified amount of years in the future, Earth has fallen to ruins in the face of the *ablockalypse* (or *ablockolypse*, the humans studying this massive issue can never decide on a spelling, and to be fair they have more pressing matters than fixing the spelling of a made-up

word). All that remains of buildings are the structural beams and pillars. Chunks or blocks of structures, pipes, and random hunks of metal fall from the sky at rapid speeds. The cat, Chirpina, thrives in this world, leaping onto various platforms of metal to traverse the terrain. Chirpina is a very sweet cat, a playful one with a constant desire to jump, convenient considering the circumstances. She may fall sometimes, but she always gets back up and continues on.

Gameplay



The world has two distinct settings that alternate between levels. The first setting is the basic platformer, where you, as Chirpina, start on the left pillar, must leap onto the generated platforms and make it to the other pillar. The second setting is similar to the first in that you are Chirpina starting on the left pillar and must get to the other side, however the middle section is not filled with platforms. The game converts to “Tetris-mode” and the *tetrominos* fall from the sky. You, the player, must guide the pieces into a formation that Chirpina will later use as her platforms to make it across. There is a timer during this part of the game. Once the timer is up, you are once again playing as Chirpina and get to use the pieces you have just guided into place as your platforms to get across.

Controls

Designed with a keyboard-player in mind, the controls are based on what felt most natural for playing for platforming and for Tetris. The initial screen has an indicator to press the **Spacebar** to start the game, and that is the only time that button is used. For the platforming portions of the game, the player should use the **Up** arrow key to jump and the **Left** and **Right** arrow keys to move. As for the Tetris gameplay, the controls are the **Up** arrow key to turn the piece, and the **Left** and **Right** arrow keys to move the location of the piece. There is also a **Down** arrow key to make it go faster but considering the speed-up, the developers do not recommend using this if you want to pass the level. For the visual satisfaction:

	PLATFORMING	TETRIS

UP arrow key	Jump	Turn tetromino piece
LEFT arrow key	Move Chirpina to the Left	Move tetromino to the Left
RIGHT arrow key	Move Chirpina to the Right	Move tetromino to the Right

- Development (Design Patterns)

Each tetromino (block) has a special shape, as it does in Tetris, and the individual designs are hard-coded so a library of the pieces could be made. While not a design pattern, the struggle of determining how to flip the pieces how we wanted was incredibly frustrating and setting them up as hard-coded arrays made it possible to implement some linear algebra concepts, namely matrices and how to change their orientation. Since each tetromino was hard-coded as an array that was then put into a bigger array, the option of making this an object instead did come to mind. The greatest problem with that, however, was that it would be an object that only held the information of the matrix in it in order to do the rotations. To contrast, the animations for the cat are infinitely looping mini game loops that we treated as a singular object and caused nearly no issues comparatively.

As expected, the update methods are packaged with multiple other function calls. Essentially, the update methods we created call on the functions that act as commands. As an example of this nesting: the TIC() function, the game loop that was known ahead of time, calls the update functions “handleModes” and “switchPlatMode.” The “handleModes” function acts as a command that switches the states of the game components that exist regardless of the game mode (such as the background) and sets the controls to the proper mode, as well as activating that gameplay. In order to do all of those commands, the function calls all of the other necessary functions for gameplay. The “switchPlatMode” plays the role of getting the first screen ready for the tic update. This is technically a double buffer since it is “preparing” the following screen, but it is more of an instant reaction to the **Spacebar** activation.

The function for controls has the equivalent of commands modifying different states that the cat character has. Since there are only three actual controls to configure, it made more sense to just hard-code it for each kind of gameplay mode. The cat has a physics portion (velocity and acceleration, and a separate directional note for ourselves) that gets changed when different buttons are pressed, and there’s a separate portion for when the cat jumps because the jumping can and should happen while there is left and right movement.

Conclusion

The game is beautiful and, as usual, the sound effects were the least of our concerns. We tried to implement the design patterns we learned about in ways that were not as “given” since we are working in Tic80. If we were to add more layers of complexity to the game, it would be interesting to see what would happen if the “row elimination” part of Tetris were also in the game. Besides the difficulty to add it in, it may make the gameplay itself more difficult.