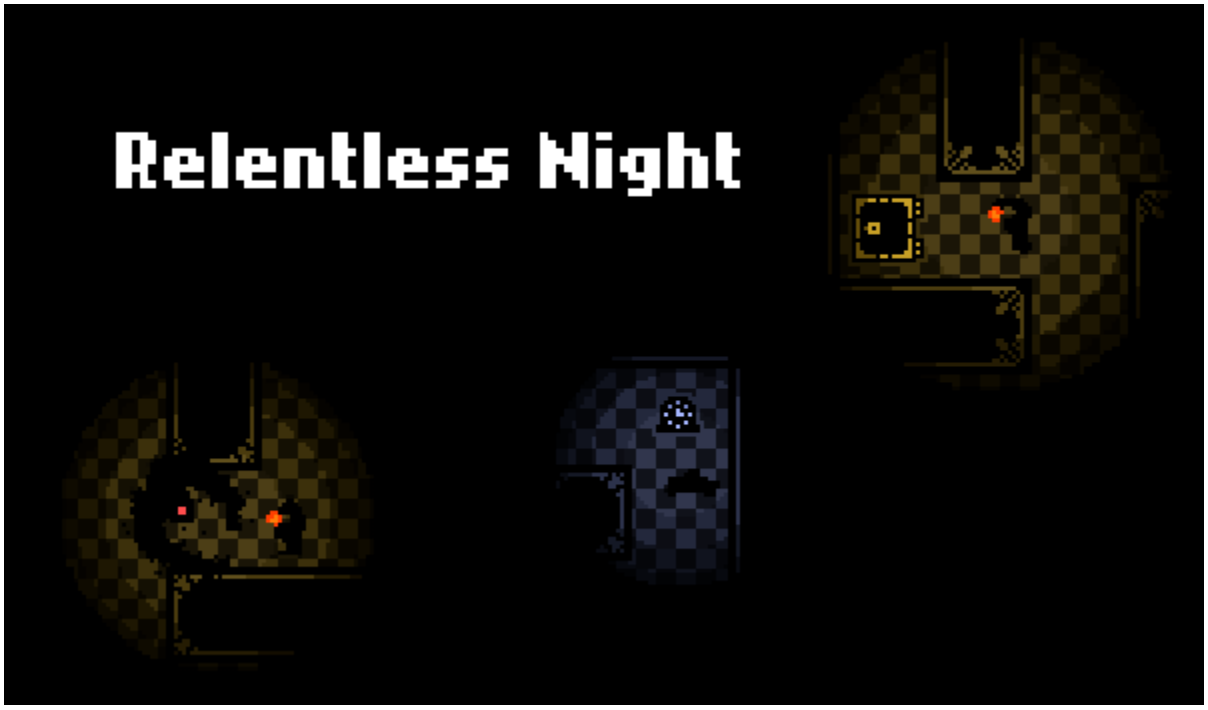


Relentless Night

Postmortem

Introduction/Overview:

The following is the postmortem for Relentless Night, a survival/dungeon-crawling/roguelike game developed by Elliot Goodzeit, Alexander Kuribayashi, Matthew Mitsui, and Jerry Reptak for the Spring 2012 Video Game Programming and Design course at Rutgers University in New Brunswick, NJ, USA. In this course, students were taught game design fundamentals, created several game prototypes, and developed a subset of those prototypes into full video games. This is one of the three fully developed prototypes for the course.

In Relentless Night, the player must navigate a character through 8 floors of large mazes, acquiring power-ups, avoiding enemies, and reaching each floor's exit. The character starts above the 8th floor and must survive the entire trek to the ground floor. If he is caught by an enemy, he dies and is sent to the beginning.

The game was intended to create a sense of claustrophobia and fear through its visuals and sounds. Our design choices, including the ones to foster this aesthetic, are listed below. Below are the most notable successes and obstacles that were encountered during the development process. As we hope will become clear, while we feel our development time and infrastructure

were not managed optimally, our choices ultimately drove us to make a simple game that seemed to foster our desired aesthetic.

Specs:

Name: Relentless Night
Developer: Team Darkness
Team Members: Elliot Goodzeit, Alexander Kuribayashi, Matthew Mitsui, Jerry Reptak
Development Time: 8 Weeks
Platform: PC
Software Used: FlashDevelop 3.0, Flixel 2.35, Flex 4.6.0, AS3 Data Structures For Game Developers (AS3DS), Flash, Audacity, SFXR, TortoiseSVN, Github
Hardware Used: Windows 7 PCs

What went right:

Choice of Flixel/AS3

Flixel, a game development library made by Adam 'Atomic' Saltsman for ActionScript3 (AS3), was used to develop this game. Flixel has several built-in functions that abstracted painstaking programming details necessary for our game (e.g., sprite sheet animation, pathfinding in a maze between two points, screen shaking, and collision detection). This made much of our higher-level programming easier. While Flixel's default functionality was not wholly sufficient for some of our desired game behavior, Flixel is open source, allowing us to create code analogous to the native code that better suited our needs.

Final Art and Aesthetic Choices

As previously mentioned, we intended for this game to, in general, evoke a sense of fear and claustrophobia from the player. We attempted to evoke this through several simple visual and audio cues. Only a small, local area around the player is lit at all times, with lightning occasionally illuminating the rest of the room. Each room, except the first, last, and "purgatory" rooms, contains enemies that make footstep sounds and whose eyes glow in the dark when pursuing the player. This game walked a fine line between tension and unplayability, as some bad iterations of visual and audio cues like these made the game boring or frustratingly difficult. Playtesting in the latter part of the semester showed that our later iterations of the game, in general, generated the desired tension.

We also chose to integrate the credits into the game so that the player is constantly immersed in the game experience. We also included a "purgatory" level when the player's character dies so that the immersion is not too overwhelming; the player can take a break while still remaining immersed.

Version Control and Task Management

Choosing git and GitHub as our version control proved to be a great idea. We created various testbed branches for developing certain tools (e.g., maze generation) without halting or breaking production of the game itself. Github gave us a convenient user interface to all our commits and let us comment on certain lines of code and create issues. In addition, Asana, a task management software tool, helped us quickly sketch tasks and delegate them to team members. Asana allowed us to write elaborate notes with respect to each task, and it has several RSS feeds that we used to monitor project progress. We used Asana to delegate tasks to the specialists who were most capable, helping to alleviate the problem of code segmentation (see "Unclear Code Segmentation" below).

Sensitivity to User Feedback

When modifying the direction of the game, our best feedback came from our users (e.g., our classmates, TA's, and professor). Since many of our non-team users were well-versed in video game playing in general, they were able to give honest, concise, elaborate descriptions of what they felt was wrong with our game's design and mechanics, offering creative solutions. User feedback was the largest deciding factor in whether features made the cut, and it ultimately helped us narrow our vision.

Minimal Content/Conceptually Simple

While much was unclear with respect to game direction, we found that focusing on a small amount of content allowed us to tune a small number of parameters to make the game successful. One example is maze generation. Mazes were procedurally generated, and enemies were procedurally placed. This allowed us to tweak a small number of maze and placement parameters to properly scale difficulty; we did not have to manually create each maze. In an attempt to alleviate user burden (i.e., mental bookkeeping) by reducing the number and complexity of states, we kept the set of powerups to a small set that is "natural" for this game's setting. In a similar fashion, the player only encounters 2 types of enemies, and the controls are also simple WASD controls.

What Went Wrong:

Lack of Unifying Vision

Instead of iterating on the prototype and beginning to improve the actual gameplay, we spent several of our first meetings deciding on the artwork direction. This caused us to produce a version of the game that strayed from the vision of the original prototype (see "Beginning Weeks" in "Art Style Changes" below). In addition, because of our lack of common vision, we spent several days experimenting with several controls and game features that did not make the final cut; some of them were made to accommodate this art style. We produced a game that starkly contrasted with the original prototype and were forced to start from scratch. The first few weeks should have been spent improving what already existed (e.g., player controls, enemy AI, and level generation).

Unclear Code Segmentation

In the beginning of the development process, we adopted a Model-View-Controller framework. The MVC framework initially split class responsibility clearly and concisely. In addition, we tried to abstract as much functionality as possible in methods and global constants. Some later functionality was unfortunately hardcoded, and the resulting code was not modular. For instance, some operations modified the enemy and player sprites directly, and others modified a separate hitbox (hidden during gameplay). The displayed sprites and hitboxes were also tightly coupled with respect to position and velocity. Some operations dealt with entity (i.e., players, enemies, and items) locations in maze tile coordinates, and others dealt with entity locations in pixel coordinates. These factors involved much mental bookkeeping during development to determine which variables to modify (e.g., the player sprite or its hitbox but not both). This made debugging difficult, as everyone would need to have extensive knowledge about most of the system.

Initial Communication and Meetings

From the beginning of the game's development through spring break, we met very little. As such, during those weeks, few ideas were communicated and exchanged about the general direction of the game, and we hence encountered the problem in "Lack of Unifying Vision". After spring break, we finally set 3 fixed days per week for meeting, resulting in much more active, productive communication. Setting these 3 times as hard constraints in our weekly schedule allowed us dedicate our full energy on those times to game development.

Choice of Flixel/AS3

While Flixel and AS3 offer several useful features, both from game development and program design perspectives, they suffer from a few shortcomings with respect to our game's development. For instance, AS3, while object-oriented, does not directly supply complicated data structures (e.g., heaps), and several graphics effects were not easily accommodated by

Flixel (e.g., shading, particle systems). Decoupling enemy and player sprites from their respective bounding boxes for collision was difficult, ultimately requiring an invisible hitbox late in game development. Unfortunately, we were also unable to have stereo sound to communicate the direction of enemy sounds.

Little Playtesting From Non-Team Users

Outside of class, much debugging and playtesting were done by the developers, and some playtesting was done by people who were not on the team. The most extensive non-team playtesting was done in class, and we found such playtesting to be invaluable (see "Sensitivity to User Feedback"). While some testing was done by non-team players beyond class hours, our ambitious goals for playtesting were not met. Several of the game's problems, whether large or small, would have been eliminated earlier through much more extensive playtesting. The lack of playtesting arose due to a lack of infrastructure for making our game easily accessible to users. In hindsight, since our game is written in Flixel/AS3, this could have been done, for instance, by embedding the game in a page and sending the link to eager, willing playtesters.

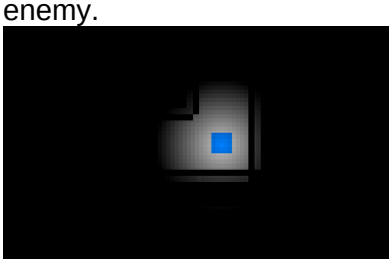
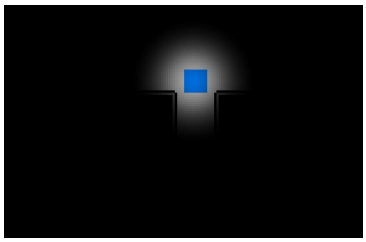
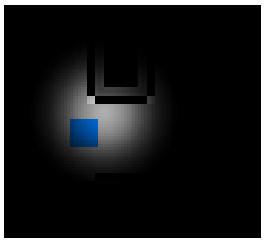
Conclusion:

What was learned from the prototype was to focus on the most important part: the game itself, not the art style. Many of the beginning weeks were spent discussing art styles and context. Through playtesting, we learned to keep our game simple and to stick to the original design set around fear and claustrophobia. Our direction became clearer in passing weeks through feedback from our peers and superiors. Better management of coding time, coding infrastructure, and playtesting methodology would have helped us converge to an optimal solution more quickly, but we feel that this game has served its intended purpose and are content with the result.

Art Style Changes

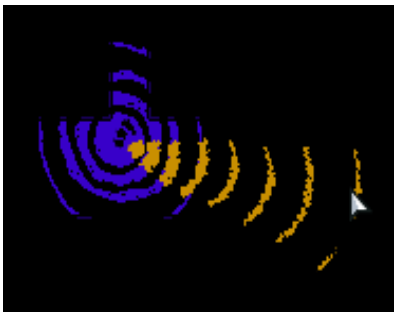
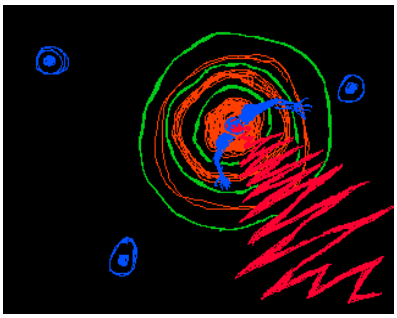
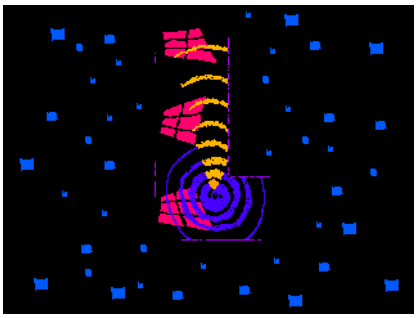
Prototype

The prototype began as just a maze with a blue square as the player and a red square as the enemy.

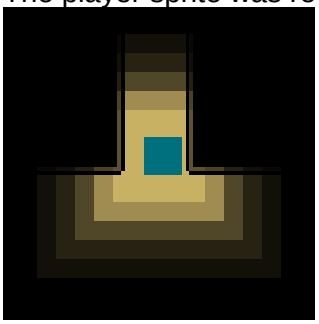
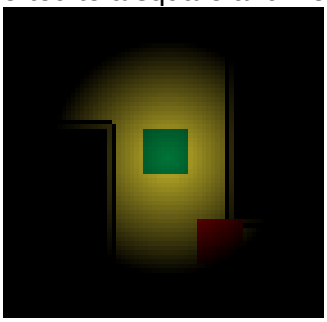




Beginning weeks

The art greatly changed between the 1st and 2nd weeks. This was one of the problems that caused us to lose valuable time for polishing existing game constructs. This art style changed the feel of the game too drastically and was reverted to the prototype's design (a box). The art style wasn't touched for another 3 weeks after this until the game became playable.

The player sprite was reverted to a square and we experimented a bit with a different lights:

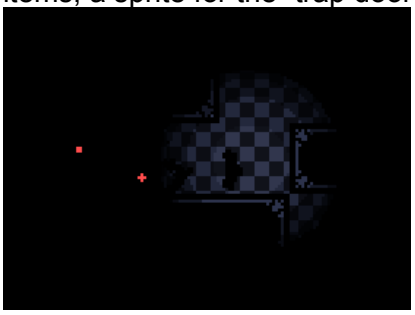



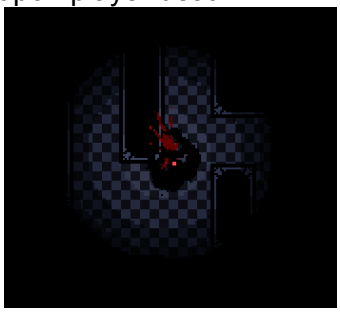
Middle Weeks

Once we established the core mechanics of our game (debugging and tweaking aside), we started working more on the art style. The player sprite was created and we worked on an orangish tint for the light. This light color was later changed to white because of suggestions from classmates. The floor was given a checkerboard design to convey a sense of motion to the player.


Final Weeks

The player sprite was animated and an enemy sprite was made and animated. Also, we added items, a sprite for the “trap door” exit, and blood splatter upon player death.





Tiles

Different tiles are displayed through level progression.