Human Harvesters

Alpha Release Report

Table of Contents

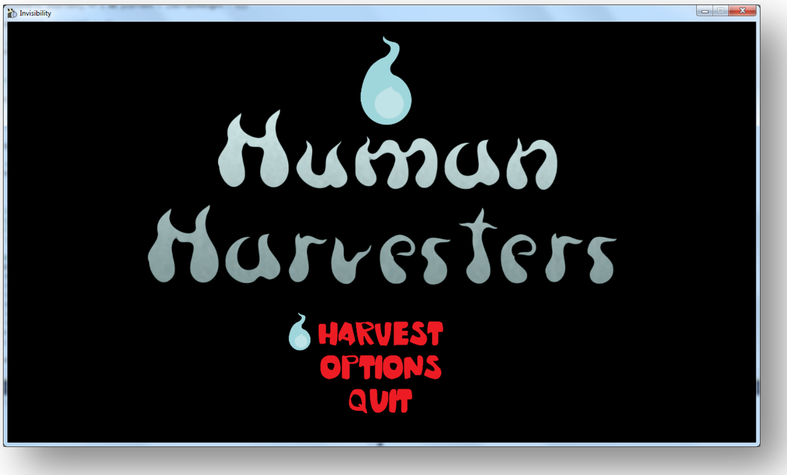
Current Progress

Development schedule

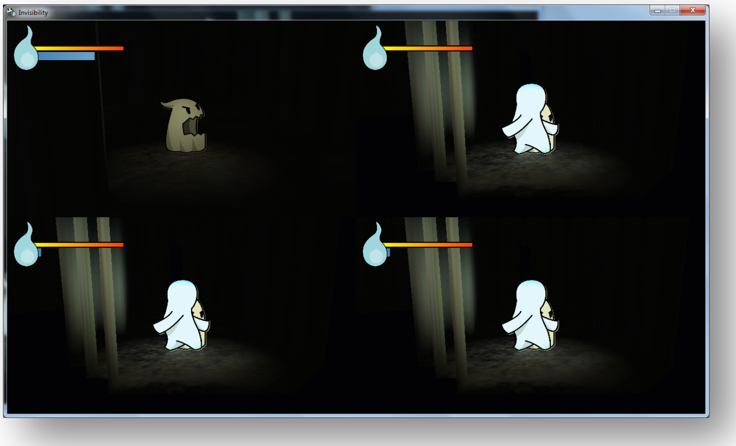
Task	Planned completion date	Status
Physical prototype	20 March	🟩
Tutorials and research	20 March	🟩
Functional minimum		
Player has an Avatar that can move	20 March	🟩
Player can kill other players	27 March	🟩
Player re-spawns when dead	3 April	🟩
Map exists	27 March	🟩
Ambient light	3 April	🟩
Low target		
Map has obstacles	3 April	🟩
Light mechanism for invisibility with time limit in place	3 April	🟩
Basic shading/lighting (deferred rendering)	17 April	🟩
Humans are on the map	3 April	🟩
Players can consume humans	10 April	🟩
Desirable target		
Interim report and demo	17 April	🟩
Texturing	17 April	🟩
Game menu / GUI etc.	10 April	🟩
Basic sounds	24 April	🟩
Animated avatars	17 April	🟩
Humans have actual models	17 April	🟩
Background music (not necessarily self-composed)	24 April	🟩
Alpha release and report	May 8	🟩
High target		
Possibility for random maps	--	🟨
Advanced stuff on map (water etc.) and effects for abilities	--	🟨
Advanced textures	--	🟩
Bump mapping	--	🟩
Multiple abilities with the corresponding effects (power-ups)	--	🟥
Advanced lighting/ shading	--	🟩

(e.g. reflection simple built-in shader / phong shading)		🟩
Advanced sounds	--	🟩
Humans run away (AI)		🟩
Playtesting & conclusion		
Playtest report	14 May	🟨
Playtest presentation	22 May	🟥
Conclusion chapter	28 May	🟥
Public presentation	29 May	🟥

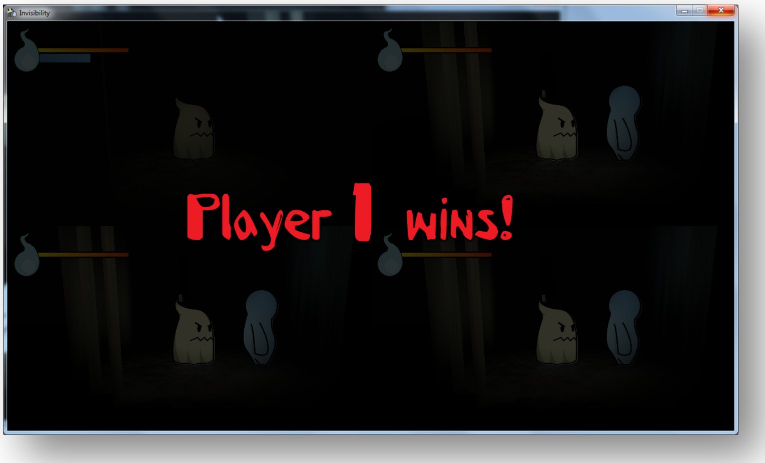
Annotated screenshots



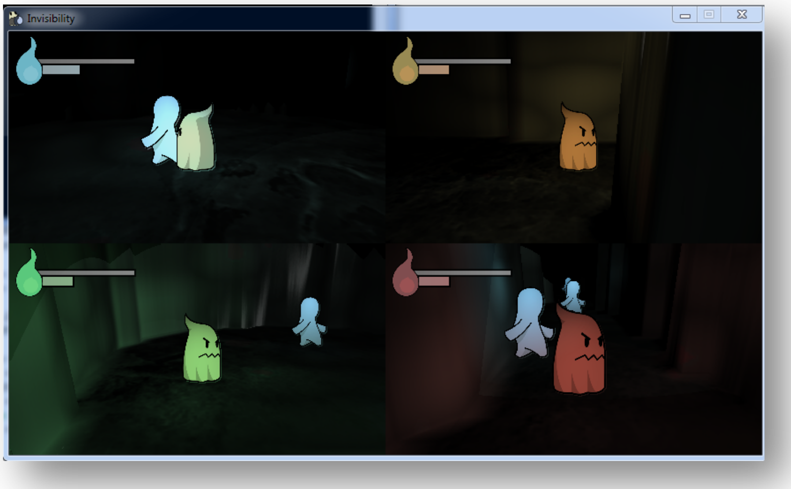
Our current menu screen, including our logo and an animated graphic. We start playing creepy background music once the player enters the game, but the playful font and logo already get the player in a fun and not-too-serious mood



Our game's look and feel before our internal evaluation. Four players have identical light colors and status bars. All of the basic player-player and player-human interactions are complete, and the humans bounce around with a simple AI.



The essential game mechanics were finished, so you can start games, and, if you performed well, win (as shown here).



We re-evaluated our graphical approach after our internal playtesting, and decided that players needed to have separate colors to make it easier to track who was who during gameplay, to add our newly thought-out dodging mechanism, and to give players another visual cue to which bar is their "life bar" and which is their "invisibility bar."

Some new developments explained

Map

The map has gone from 1 file to 2 bmp files. The first file is a simple height-map that includes information on what parts are impassable. If the value of the "red" color is higher than the other 2 values for a pixel, then that part of the terrain is impassible.

More interesting is the 2nd bmp file, the "info map". This contains the spawning places for players and humans. It also contains the locations of directional lights shining from above. At the moment, these "ceiling lights" have not been tweaked (light intensity for example), and it acts almost the same as ambient light. These are marked with red, blue, and green, respectively. The color "white" is used to mark walls, making the placement of the points mentioned above easier.

The red-black image is the height-map; the one on the right is the info map.

"Sliding" along walls

The collision of the players with the walls was changed. Previously, if the player collided with a wall, he simply wouldn't move. Now he slides along the wall, and the speed at which he does this depends on his angle to the wall. This makes navigating around the map in tight spaces significantly easier.

This was implemented by moving the player first in the x-axis direction, then in the y-axis direction. The "forward" and "right" movement of the player (read from player input) are projected onto the X and Y axis using the dot product to get the speed in that direction. For example,

```
X-axis_speed =
    dot(Forwardvector*forwardspeed+Rightvector*rightspeed, (1,0,0) )
```

This would be the maximum amount of distance that the player would be allowed to move in this direction. The player is then moved (not so efficiently) as much of this distance as possible. The same happens in the Y direction.

The reason we chose this method is because almost all the walls are at least partially axis-aligned (each pixel in the height map corresponds to exactly one vertex). The method does however seem to work fairly well for diagonal walls as well.

Easier Than Expected

One aspect of the game development that we were pleasantly surprised to find easier that we had expected was the development of the sound effects. Using fairly basic tools (Adobe Soundbooth found in CS4) we managed to record sounds effects from our voices and slightly alter them to give clean-sounded effects to the in-game actions. XNA played very well with the audio files, and though it bloated the size of our project significantly (from < 1 MB to ~15 MB after compression), there were no major hiccups with integration. Also, finding a couple of extra sound effects and some background music online that we were free to use was easy with a few basic Creative Commons searches.

Another part of our development that has been going surprisingly smoothly is collaborating together as the size of the project increases. We had originally thought that because we sacrificed purity of design for ease of coding, we would struggle to add features as our code base grew. However, it has proven to be quite simple to add in additional elements to the game, which speaks to the size of our project (still relatively small).

Harder Than Expected / Implementation Challenges

We have not anticipated getting the lighting to be correct being so hard. We imagined that once we had gotten the deferred shader written and could manage light sources we would have no trouble tuning the parameters. However, it has turned out to be quite difficult to let players see ***enough***, so that it's fun and easy for them to navigate the map, but ***not too much***, so that they can't spy other players or prey from a far ways away.

We also didn't expect our map to look so lifeless. This is something we're not sure how to fix, given we have a map generated from a height-map (rather than a pre-modeled one). We want to make it look more interesting and cave-like, but it's hard to get the look right while maintaining the closed-off and dark feel that is essential to the gameplay.

Design Revisions: Pre-Alpha Playtest

One week before our alpha release, we all got together to have a mini-tournament of our game in order to evaluate how it played and what the top-priority items were to fix before the alpha release came out. We made a list of everything we noticed that we wanted to change, and this became our first priority to-do list for the next week. Here are some of the things we saw:

- Fix for 3 people playing (players appear in wrong order on screen)
- Fix for 2 people playing (e.g. players 2 and 3) (incorrect controllers selected)
- Fix the winning text on even numbered games
- Add sliding along walls
- Show text when you die
- Bar reaching soul marker = dead (right now some wiggle room)
- Animated soul icon when close to death (e.g. flash white)
- Invisibility bar is white/transparent
- Different color soul icon, soul bar, and light (source) for each player
- Multiple re-spawn points (colors on the bitmap?)
- Humans have to be placed not in walls (colors on the bitmap?)
- Humans must re-spawn between games
- Add at least one more map
- Show targeted player (for example, soul-drain button above them)
- Show targeted ghost (for example, kill button above them)

- Change “Draw!” text to say “You all fail!” or something (because we said there would be no draws…)
- Lantern for ghosts (3D models)
- Torches for walls (3D models)
- Rocks for environment
- Put bounds on the brightness / darkness changing ability to avoid graphical glitches
- Add sounds for eating humans
- Add sounds for draw
- Add sounds for escape mechanism
- Make escape mechanism only available when under attack
- Make it so that you can’t turn invisible while being attacked
- Add directed light

Escaping Mechanism

- When a player is attacking, the controller rumbles (and warning text appears on screen), and you can “Color press” -- you must press the color button of the player attacking you. If you dodge successfully, your light emission flashes very large, you dodge any current attack, and there is a cool-down afterwards.

This process was extremely valuable in order to re-evaluate our priorities from the perspective of gamers and not designers, and we set this as our to-do list for the next week.

Playtesting Goals

1. Get feedback on our **map** design, and continue to improve our maps until we get positive feedback that people can navigate well.

2. See if people use the main **game mechanisms** we hope they will (becoming invisible and dodging attacks), and continue to add hints and cues and rewards for these mechanisms until people use them more readily.

3. See if people play the game using **strategies** we hope they will, and tune the game to reward our intended use cases and punish others (e.g. reward sneaking, punish brazenly running and fighting).

4. Get overall **aesthetic** impressions from people about the quality of the sound, visuals, menus, help, and **experience** of playing the game.