Game Metrics Without Players: Strategies for Understanding Game Artifacts

Mark J. Nelson
Center for Computer Games Research
ITU Copenhagen
Copenhagen, Denmark

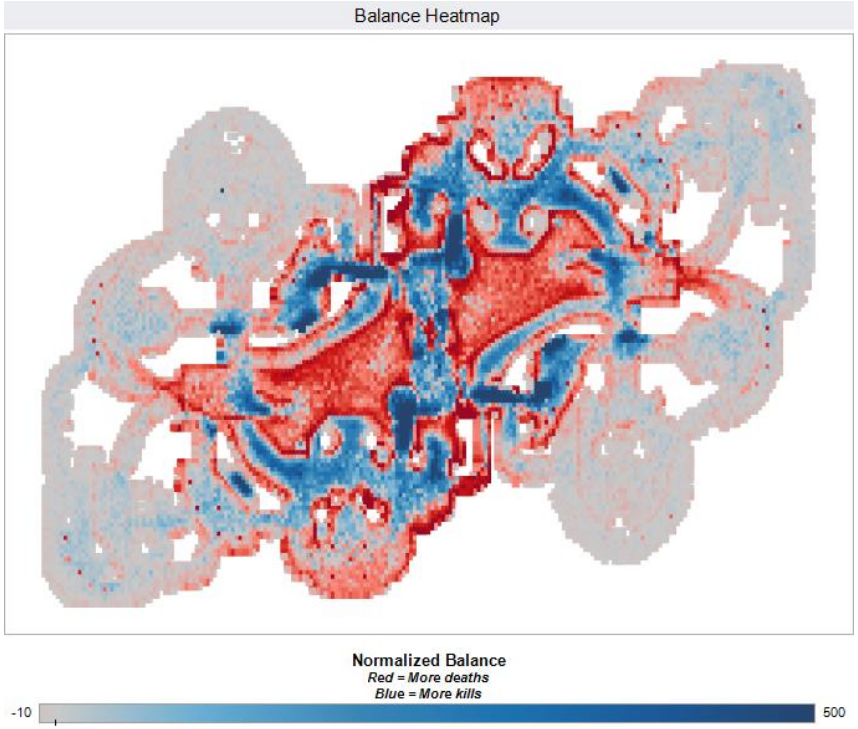
Keaton Schwartz

Introduction

A common way to get empirical data on your game is through playtesting. There has been a lot of research on ways to use AI to better visualize data from playtesting.

Introduction

Use heatmaps to show player data



Introduction

In these tests, the source of information is *exclusively* the player. This is a great way to understand how your game is received, and understand the *game artifact:* the system of code and rules.

“Can we use AI and visualization techniques to augment that thinking-about-the-game-artifact job of the designer, the way we’ve augmented thinking about player experience?”

Strategy 1: Is this possible?

Strategy 1: Is this possible?

Ask “Can X happen?” questions.

Is this game winnable? Can the player beat the game without taking damage? Can the player get stuck in this room?

Answers yes/no questions whose answer is determined solely by the game rules.

Strategy 1: Is this possible?

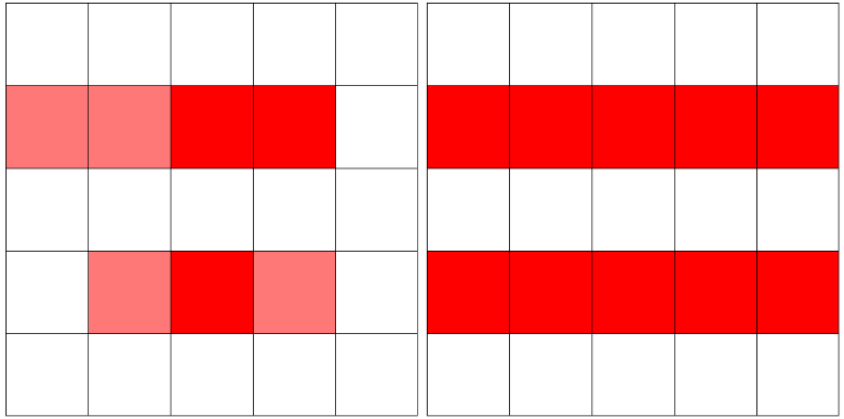
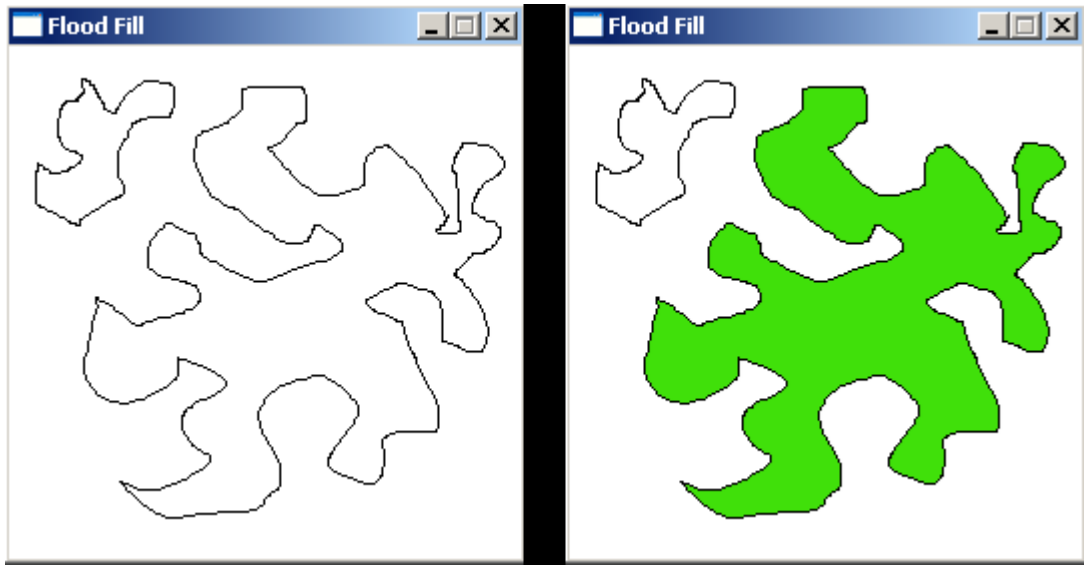


Figure 1: Heatmaps of player deaths: left is empirical deaths from several playthroughs, while right is analytically possible death locations.

Strategy 1: Is this possible?

Floodfill



Strategy 2: How is this possible?

Strategy 2: How is this possible?

If we want to know how something is possible, we often keep a log of player actions.

```
happens(move(player,north),1)
happens(attack(monster,player),2)
happens(attack(monster,player),3)
happens(die(player),3)
```

Figure 2: Gameplay trace of a player dying.

Strategy 2: How is this possible?

These logs are often pretty boring, we only care about the “interesting” ones.

```
happens(move(player,north),1)
happens(attack(monster,player),2)
happens(attack(monster,player),3)
happens(die(player),3)
```

Figure 2: Gameplay trace of a player dying.

Strategy 2: How is this possible?

For an outcome that leads to a game state that should never happen, the log will always be interesting. To find other “interesting” possibilities, let the designer refine the trace they’re requesting.

For example, we could ask for a way the player dies while standing around doing nothing, and then go on to add more caveats if the result were still too mundane; we refer to this as trace zooming

Strategy 3: Necessity and dependencies

Strategy 3: Necessity and dependencies

What must happen?

“Can you beat *Final Fantasy IV without ever casting ‘Meteo’?*”

What quest must be completed?

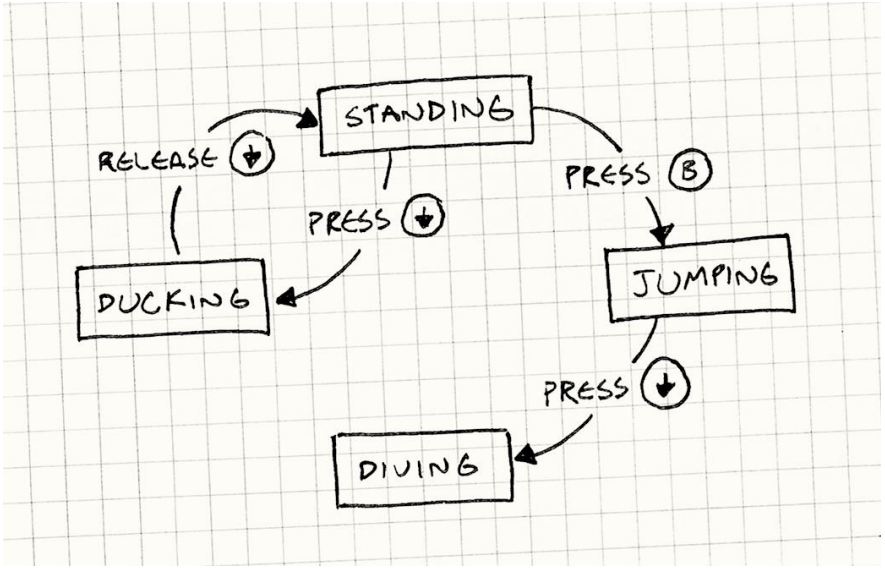
Is getting the master sword necessary to beat the game?

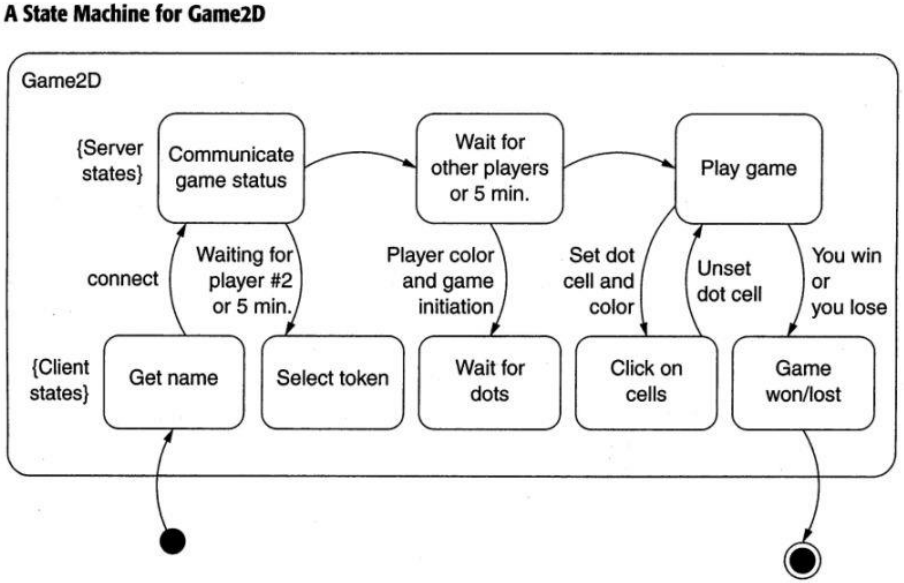
Strategy 3: Necessity and dependencies

“asking whether a particular sword is necessary to beat the game is equivalent to asking for a gameplay trace where the player beats the game, which doesn’t contain the pick-up-that-sword event.”

Strategy 3: Necessity and dependencies

Make a graph to determine event structure and possible game states





Strategy 4: Thresholds

Strategy 4: Thresholds

Is there a magic number delineating the bounds of possible behavior?

How fast can you beat this level of *Super Mario Bros.*?

How much money can you get in the first 5 minutes of *City Skylines*?

Strategy 4: Thresholds

Answer things that may not be possible in an initial playtest because they would require speedrunning and optimization.

Strategy 4: Thresholds

Use these distributions of min and max times to better understand game

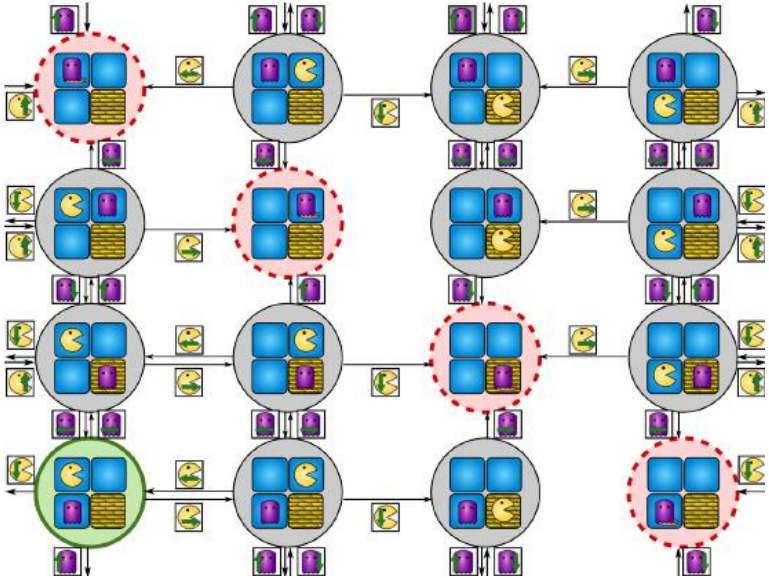
Strategy 4: Thresholds

Also algorithms that are possible to implement to try and understand this. Use Dijkstra's Algorithm to determine the shortest spatial path through a platformer.

Strategy 5: State-space Characterization

Strategy 5: State-space Characterization

Understand different state-spaces. It would be impossible to show all possible game states, so instead we can look at just one trace of the game state graph and then look at neighboring states/traces to understand how to achieve a certain condition



Strategy 6: Hypothetical Player Testing

Strategy 6: Hypothetical Player Testing

Show how game would work in extreme/highly idealized cases. This could represent a simplified model of a player.

For example, what happen when the game is played by a player who always attacks when seeing an enemy, then heals when there is none?

If the player does very well, the game may be too easy.

Strategy 6: Hypothetical Player Testing

Testing using the machinations system: “applies strategies 1–4 conditioned on a player model, answering various questions about possibility, necessity, etc., under the added assumption that the player is acting in a particular manner.”

Testing using the Monte-Carlo “rollouts” of boardgames: pit two different strategies against each other at different points and determine who fares better

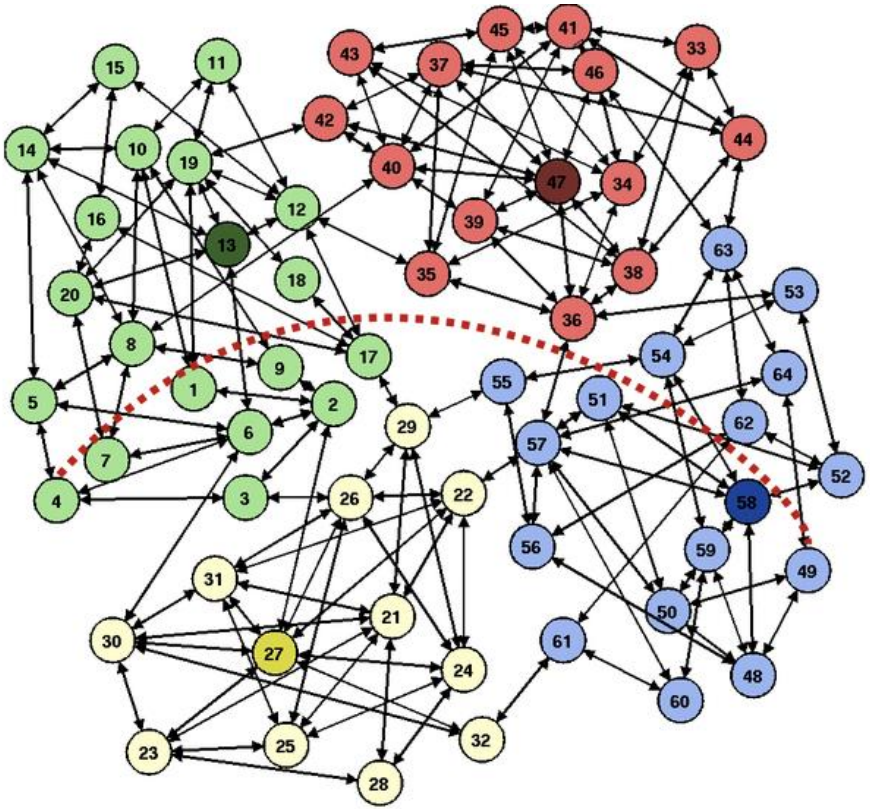
Strategy 7:
Player Discovery

Strategy 7: Player Discovery

“Discover” different types of players

To do this, look at several different traces of game states that end up at the same condition, analyze the different decisions made to reach this state

Strategy 7: Player Discovery



Strategy 7: Player Discovery

For example, how did players beat the first boss?

Player 1: Got weapon, upgraded health, recruited help, beat boss

Player 2: upgraded health, explored world, got armor, killed by boss, beat boss

Player 3: recruited help, set up house, got weapon, beat boss

Conclusion

Conclusion

Not meant to replace playtesting

Instead helps developers to better understand their game

“the game artifact itself need not be a black box, since we can learn many things about a game design simply by better understanding the game itself”

Conclusion

Nelson hopes that “This analysis of games can benefit from an ecosystem of metrics and visualization techniques that will hopefully grow as rich as that now featured in player metrics research.”