VENUS

Erick Bravo | Jazlyn Fernandez | Juan Vazquez
Angelina Nantanapramoth | Daniel Esquivel | Kathareeya Atthajaroon
Department of Computer Science
California State University Northridge

Abstract

When playing a video game, there's often a sense of thrill when players are out exploring in the digital frontier. However there are moments in games where one is often left confused on what to do. Usually these confused moments are met with hints and guides to get a player out of that situation. That is, if it's added to the game. New video games such as *Scorn* offer visually fantastic games but they leave the player wondering what exactly is going on. There is no built-in sense of what to do and if the player does manage to figure something out, they wind up frustrated at it having taken so long. What we aim to do is create a virtual helper that will calm down those moments of confused panic. The helper, Venus won't play the game but will be a vigilant sentinel ready at a moments notice to leap in and see what the player needs help with gamewise.

Objectives

- Create a machine learning assisted hint system called "Venus".
- Predict when the player is frustrated with the following methods:
  - Capturing the player's cursor acceleration.
  - Counting the number of clicks the player has made before they've completed a game objective.
- Once the system determines that the player is frustrated, the player is asked if they would like to activate the hint feature.
  - Hint feature consists of a "Hot or Cold" game, signaling "Hot" when they are closer to the game's next target goal and "Cold" otherwise.

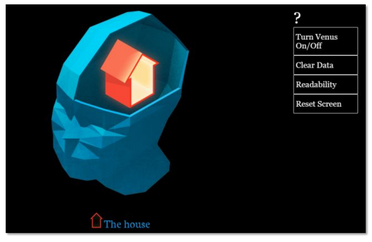


Figure 1. Venus user interface integrated to work with "The house"[1] by Artur Kot. The Venus interface is composed of the question mark element on the right along with the white boxes and text underneath the question mark.

Figure 2. Venus user interface asking the player if they need help while playing "The house"[1] by Artur Kot..

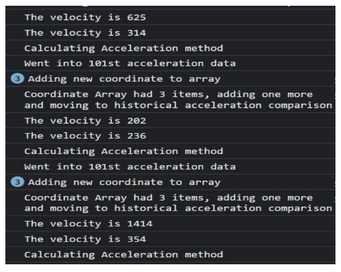


Figure 3. Real-time example of Venus's calculations of the mouse accelerations

Figure 4. Test sample of Venus predicting player's frustration and reading game progress

Too often games come out with complex tasks that seem obscure to the player. Our Venus Hint system, assisted by machine learning techniques, is specifically designed to alleviate gamer's frustrations caused by the complexity of the puzzle game. Our goal is to provide assistance to gamers during gameplay, while also respecting their preferences by allowing them to disable the hint system. We want Venus to give value to the player by allowing them a way to play the game without unnecessary frustration. Venus will help with that undue stress. This will give a sense of satisfaction when the player completes the game.

Methods

- Venus collects data from the player's mouse movement; the number of clicks and a historical array of calculated mouse accelerations. *(see Figure 3 & 4)*

- In the event that the number of attempted clicks surpasses the established threshold, or there's a high differentiation between the current acceleration and its historical data, it would trigger the system to prompt if the user requires assistance. The player either responds "Yes" or "No", and resets the clicking counter to zero upon choosing.

- If "Yes", Venus reads from the game storage to analyze the player's progress with the game. This includes the items the player has collected, items that have been used, and checkpoints that they've already achieved. *(see Figure 5)*

- Venus makes list of the 'target' coordinates from the original source code so as to development a linear outline of the game overall pathline.

- Then, Venus will compare the player's current progress with the remaining uncompleted game objectives and signal to the player whether they are close (*hot)* or far (*cold)* from the next 'target' based on the player's position in the tile-based game.

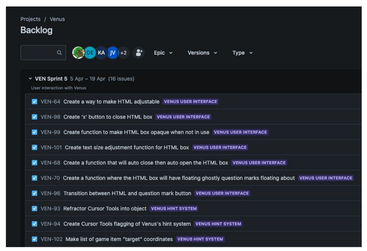
```
//Mouse Accleration
function calculateAcceleration(velocity) {
  if (previousVelocity !== null && previousTime !== null) {
    let velocityDifference = velocity - previousVelocity;
    let timeDifference = (currentTime - previousTime) / 1000;
    acceleration = Math.round(velocityDifference / timeDifference);
  }
  previousVelocity = velocity;
  previousTime = currentTime;
  return acceleration;
}
```

Figure 5.(above) Sample of our code for function that calculates the acceleration of a cursor-movement event.

```
var collected = $.jStorage.get('collected', []),
    used      = $.jStorage.get('used', []),

    played    = $.jStorage.get('played', []),

    is_in     = $.jStorage.get('is_in'),

    fish      = $.jStorage.get('fish', []),

    path      = $.jStorage.get('temp_path');
```

Figure 6.(left) Sample data.js code that Venus will utilize when hint system is activated. Venus will be able to determine what items have and have not been used, the player's position, and basically get an understanding of how much progress the payer has made.

Project Management



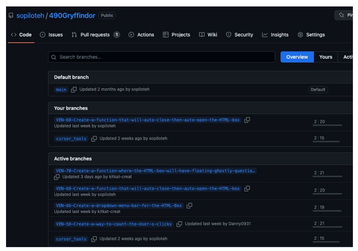


Figure 8.(left) Using Jira, we established a workflow that allowed us to transform ideas from meetings and discord discussions to fleshed out user stories and tasks. Each of these units were attached to a sprint for time management.

Figure 9.(right) Code to be tested, refactored and approved of was uploaded to GitHub. Each member of the team could create branches to implement a new function or to refactor a problem into a working solution.

Workflow

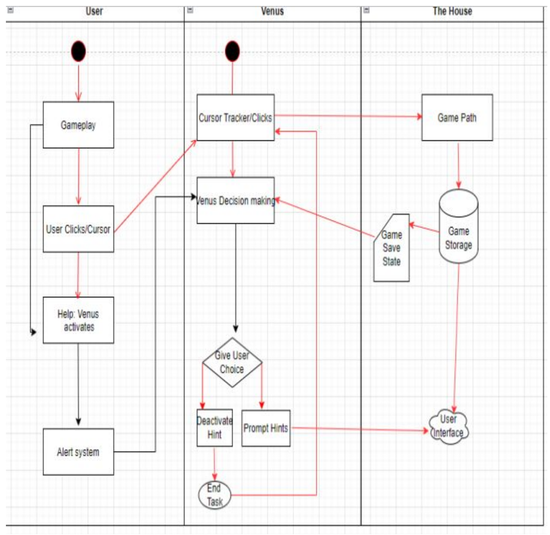


Figure 7. This is one of the many documents we produced while trying to plan this project. It explains the flow of the system: starting from the User, Venus is performing calculations on their cursor movement and clicks. If the calculations exceed the threshold, the user is asked if they need hints. If they need help we read their game data and provide hints to the next goal in the game. Once their next goal is achieved, Venus is deactivated and waits until the user needs the hints again.

Conclusion

This senior design project is a prototype of a machine learning game helper providing a working scaffold for future iterations. Core features include: methods of using mouse data to predict when the player is frustrated, reading the game state, & the Venus interface. While this shows a development version, the proposed hot & cold hint system is under development.

In the future, the project can continue with these foundational functionalities and add a hot & cold hint system. Future iterations would research and implement machine learning algorithms that predict player frustration.

References

1. "The House", a video game by Artur Kot (GitHub)
2. "A Fluid Human Interactive Proof in Virtual Environment", Li Liu & Wen-Chin Hsu (*IEEE SMC 2017*)
3. "Automated Artificial Intelligence (AI) Control Mode for Playing Specific Tasks During Gaming Applications", US Patent, Sony Interactive Entertainment Inc.(2021)
4. "Sony has patented an AI that can play video games for you", Sony Interactive Entertainment Inc. (2021)

Acknowledgements

We would like to acknowledge the following for their help in making our project possible:

- Artur Kot - game creator of "The House" used in this project
- Sony Entertainment - inspirator of AI helper idea
- Dr Li Liu - 490/4901L professor and his research paper (reference #2)