Lifeless:
Video Game's New Era.

Written script for the video talk.

Project developed by:

Luis Chacón

Contact:

GitHub – LinkedIn – Itch.io

Contents

Lifeless:

Hello and welcome to Lifeless, Video game's new era. Or how can AI change how we play our games.

This is a little project that I did to test how integration with AI of NPCs in games can potentially improve the player experience exponentially and add a lot to the immersion and expansion of the world without costing many resources in development.

As part of the presentation, the written version of this talk, as well as the presentation itself, will be available on the GitHub of the project if you want to take a deeper look at them. Be aware that the written version might have changes, and it was meant as a script rather than an investigative document.

About me:

I am Luis Chacon, a game dev by passion, however I study systems engineering and work as a Microsoft support engineer with a focus on app services.

Please feel free to reach me out on LinkedIn if you have any inquiries.

Background:

Predefined Inputs and Outputs.

Currently, we use a predefined input and output system that doesn't give much depth to the interactions with the game. We are in most cases provided a set of predefined dialogues and have a set of possible responses for it based on player stats and relation with the character. This of course in the most RPG type of games where immersion is taken as a primary measure of success.
In most cases for other kinds of games, the interaction doesn't even go that far and is plain and almost generic.

Little to No Changes Based on User Input on the World.

As is usual in almost any game, the NPCs don't react to the environment changes and when they do is in a very limited way. A great example of this is how when you kill king Radovid in The Witcher 3, there is no noticeable change on how the kingdom guards don't react to the event and keep acting as if he is alive.

Of course this can be programmed behavior and integrated in the possible states and dialogues the characters can use, however, it is a very heavy task to do as every dialog has to be manually crafter, this takes developers time, and ultimately costs money.

Current Limitations:

Scripted responses

As I mentioned, the drawbacks include a scripted nature to the characters that can sometimes partially or completely break immersion on the game. However, the effects are not limited to this and if we look deeper can be expanded upon.

Limited Behavioral Complexity

What I mean with this is that, as characters don't have a real way of thinking by themselves so to speak, they cannot react or change their behavior in a meaningful way to user inputs and changes that affect the world as they don't have context or capacity to generate new dialogues based on new stimulus.

Inconsistent Reactions

Since they don't really have any memory of what you did previously, responses can be very extremist in nature, as you can do something mean to them but since your likeness with them is too high they won't react accordingly. This causes discrepancies on how NPCs react to you and is a system based only on points but not on context, as in a real situation you might do a prank to someone and not break the friendship, but they won't just turn a blind eye, instead they might joke with you about it.

Lack Of Personality

Since they don't really have a way to change their responses, they don't have any real personality of course, this means that they are limited in terms of how they can act and will

move and do pretty much the same as all characters unless you make them have specific behavioral states.

How Can AI Help:

Here is where things start to get interesting, with current technologies there are already a lot of little things that can improve the immersion of a game, and we will discuss how this can be expanded upon in the future, developing technologies that are specifically made for games to take full advantage.

Dynamic Dialogue Generation

Of course, the most obvious way to implement AI currently is to use an LLM (Large Language Model) to generate dialogues dynamically at runtime for the characters, this single change can provide a lot of features and advantages to games that are currently not possible in an efficient and cost-effective way for companies.

Adaptive Behaviors to Context Awareness

With the help of AI we can give the characters the ability to understand the context they are immediately in, and have scope of the bigger world picture, which results in more complete and immersive responses to not only the environment, but ideally user input.

Memory System

As an inherited trait of LLMs, the characters will be able to keep a memory of past events and even have special memory for things the player wants them to remember. This provides a level of immersion never seen before as characters not only have an understanding of the world, but can also learn from the player and role play along with them to increase the sense of a living world.

This can also be integrated fully with the already standard points and stats systems adding whole new level of complexity to the interactions with NPCs, providing not only context of the world but of the relationship between the players and Lifeless

Emotional Responses

With both advance AI systems and basic points and stats systems, relationships between players and Lifeless can become a lot deeper and rewarding. Borrowing The Witcher 3 as an example again, relationships between love interests as Triss or Yennefercan be very ephemeral. Yes, they do have specific paths and quests that add deepness to them as a whole, but outside the predefined path, there is no real interaction with them. If you try to talk with them outside scripted events, they will provide very generic responses. With AI, these interactions can be more complete and add a lot to interactions outside scripted events, which is not limited to romantic interests, but can be expanded to other actors as merchants or friends of the player.

But How Does AI Work:

Quick Overview

A succinct and general way to describe an AI, specifically based on neural networks, is as a sequence of matrix functions with X and Y inputs that return a Z output. In this description, X represents the user input, Y represents the model's parameters (weights and biases) and Z the generated output, which can vary depending on the kind of AI.

Of course this is an oversimplification for the sake of easy explanation as the focus of the talk is not on how AI works but rather how it can interact with games and improve them.

In a nutshell, we can say that an LLM predicts the most likely next word that will be in the sentence, taking into consideration context and learned behavior.

X, Y and Z Representations

To understand a bit more, X is the user input on the model, this can be an image, text, etc. depending on the kind of model that is being used. For LLMs like the one used here that are text-to-text based, the user inputs a string that gets tokenized and embedded so that it can be represented in a numerical way for the AI.

This is passed along for the model to process along with the weights and biases of the model, represented as Y and finally returns a Z output. Again this is an oversimplification and I would advise you do research on AI as a whole since it is a subject worthy of talk by itself.

Overview of Lifeless:

Lifeless came to be as an experiment and test on how AI can improve our interaction with NPCs or actors in a game, after playing around for a weekend with some local LLMs and role-playing characters.

The name Lifeless comes from a book of the famous author Brandon Sanderson, Warbreaker. In this context, Lifeless are “Mindless Manifestations in a Deceased Host.” Since NPCs in video games are not alive obviously, I thought it appropriate to name them Lifeless when we give them a kind of sapience using AI and thus giving them the ability to “think” by themselves.

Being an early prototype and proof of concept, this was not meant to be a full on game (although the thoughts on how we can expand this prototype are tempting, and I might at some point add more stuff to it) but rather a quick project to test how these two technologies would hold together.

Main Goals:

Although there are a lot of ways we can implement AI on games and the thought certainly crossed my mind to add complexity to this project, the basic idea of the prototype was to offer a little kind of sandbox environment where the players can interact with created characters and have conversations as if they were NPCs on a real game to simulate the feeling of doing so in a real environment. As such, the main bullet points of this project were to have:

- Unscripted Characters
- Conversational Memory
- Character Based Personality
- Emotional Response to Player Inputs

Development and Main Technologies:

Ollama

The Ollama framework serves as the back end for all the AI processing and text generation. For this experiment, I mainly used the Llama 3 model, however, this can potentially be used with any model, and could even work with more tailored solutions suited for role-playing.

This also ensures that almost anyone can run the project, as the model can be changed to suit the machine used if it is not capable of running a more demanding model.

Godot 4

Being an OSS engine, Godot offers a lot of flexibility and for this project it enabled a very fast and dynamic prototyping. Since it has built-in functions for path finding and Plugins, this helped speed up the general aspects of the project like movement and sprite management for Lifeless.

In this case, a plugin called LPCAnimatedSprite2D was used to help add skins or textures to the Lifeless easily during the prototyping and later I chose to stuck with it for the simplicity it offered to change textures at runtime without worrying on handling sprites data.

As mentioned on the plugging name, the sprites used for Lifeless come from the Liberated Pixel Cup of Open Game Art, which is a very flexible set of sprites that anyone can use freely under the CC BY-SA 3.0, which made them a perfect match as I didn't have to worry on creating art, and it adds an immense level of customization for characters, which was at the core of the project, allowing players to customize their lifeless as desired.

JSON Files

This works to store the data of the Lifeless as the name, surname and descriptions, this could potentially also be used to hold memory between playing sessions, however, this functionality is not yet implemented.

HTTP Requests

This is the way we interact with the back end, by sending the parameters to the Ollama server using their API we can generate multiple requests and return the necessary dialogs for the Lifeless.

What is a lifeless:

A Lifeless can be described as a node or object that contains a sprite for its skin, a JSON file for its basic information and can be used as a puppet for other nodes as a player and NPCs to represent a specific character in the simulation or game.

Components of a Lifeless

JSON File

This keeps the Name, Surname and Description for the character personality. It has a specific format that it must follow, failing to have one of the key values will make the lifeless to be incomplete.

PNG File

This is the sprite sheet used as a skin for the Lifeless. As mentioned, it is based on the LCP Sprites from Open Game Art.

Note: All user created Lifeless must be stored at “%APPDATA%/AmeNoHi/Lifeless/Lifeless Skins”

How Do They Think:

So the process of connecting with Ollama is done through the API that it has available, in this case using the Generate Completion endpoint. This expects certain things to be passed in the request. In this case the parameters used are the model, the prompt and the context, all of which get passed along when the signal to call this method is emitted. After Ollama makes the completion, we handle the request result and pass along the body if the request was successful.

A thing to note is that streaming is disabled on the parameters, this is normally used so you can see each word being generated and also to let the user know the request worked and provide input while it is being completed, the behavior is most useful on big requests. The decision to disable streaming is mainly because Godot 4 uses a special node HTTP request, which doesn't support streaming of requests, it only returns the packets after the requests report completion, so it wouldn't make any difference to have it enabled.

Flow of interaction:

With all that I have explained up until now I think it's pretty clear what the flow of an interaction is like, however a quick overview is as follows:

Step 1

The user interacts with the Lifeless, in this case this is done by getting in close range to the lifeless and clicking on it, this will bring up the Chat UI and provide an interface for the user to interact.

Step 2

On step 2 the data of the lifeless is loaded in the Chat UI, this is done behind the scenes for the user and is obviously to pass along with the request. At this stage, we also check if there is a context to the conversation (aka Memory), if there is not then an initial message is composed to give the AI context of who they are and the world setting, along with any other context necessary.

Step 3

Pretty self-explanatory, as we saw last, the request is made using the HTTP Request node and basically just passes along all the information to the backend.

Step 4

Ollama takes over the process at this point and generates a completion based on the inputs provided, not much to explain here as for this talk, Ollama is a black box, we just have to know it takes an input and returns an output.

Step 5

Once the response is back it is displayed as a chat message for the user to read and respond to if necessary or simply walking away and taking back the conversation at another point.

Live Demo:

Debugging Menu

Basically, this is the “Main Menu” of the prototype, which allows you to go to different modules I created along the development of the project. The project was developed in kind of trial and error way, so I first started with the Test AI module, which was just a basic “conversational” module to test connection with Ollama and learn how to make the requests.

Test AI

As you can see there is nothing much here as what I needed was a basic input and output interphase. You can select the model [Change model to Llama 3], but you can't change the address or port of the API, which means this section will only work with the default Ollama settings.

After the fact I added an AI generated illustration to make kind of like a chatbot, but of course it serves no real purpose at all as I am not passing any kind of personality on these requests.

Just for the sake of the video let's test the functionality, I will send a simple prompt and let's see if the requests succeed.

Input:

“Hello, my name is Luis, how are you?”

This little loading animation was added because I felt the responses (especially long ones) took too long to display as streaming is disabled on these requests as detailed previously.

As we can see it does return a basic response, in this case using Llama3 as it is the model we selected.

Not much more we can see on this module, so let's go back to the debug menu, to can do this at any time by pressing PgDn. Just be careful that it will close the game if you are on the debug menu already.

Lifeless Creator

Now let's move to the Lifeless creator module, this is one of the most crucial modules as of course it provides the customization we are looking for and which is the cornerstone of the project.

There are some things to note here, of course the most obvious parts are the name, surname and description fields that add the personality to the characters. Here you can do literally anything, you can add anime characters, movies, other games, historic figures, the sky is the limit. (As long as it happened before the training data for the model was cut)

In this case, just for illustration, let's make a testing character and later on i will show you an already made couple of characters that can work as examples of the extent this tool can work to give uniqueness to your characters.

Values:

Name - John

Surname - Doe

Description - I am John Doe, a placeholder, and I don't exist in this universe. I should not interact with players, or I will break the reality itself.

A very crucial part of the creation is, of course, to give them an appearance that looks as the character we want. To this extent, I decided to use the great tool Universal LPC Spritesheet Generator. In this tool you can really customize your character a lot, I did make a custom skin for John here, just to have as an illustration on how this works.

Universal LPC Spritesheet Generator

Once we have the skin, we should export it as a PNG file and save it with the format [Name]_[Surname], this can then be placed on the Lifeless folder. Just click on the Open Lifeless Folder button and place the PNG file here, once it is there you can click on the Reload Skin Button to see the skin applied and shown in game. You can also change the animation here with the dropdown next to the Reload Skin button.

Once we are happy with the changes, make sure to click on the Save Lifeless button, this will create the JSON file at the needed location with the data provided so it can be loaded later.

As mentioned, all user created Lifeless are stored on:

“%APPDATA%/AmeNoHi/Lifeless/Lifeless Skins”

Lifeless Viewer

Using the Lifeless viewer we can basically check if the Lifeless was correctly saved, if it shows here, then it was. This module is simple since at the time I only needed to see the textures applied and change animations for testing, so there is not much to do here. Just select a Lifeless and hit the Spawn Lifeless button to see the animations.

Test Movement

This module is probably the most complete there is, as here is where I tested movement and real chats with characters, let's hop in and see it in depth.

As you can see we have a very simple character and a viewport, of course you can move the camera around by using left-click on the world, and zoom in and out with the wheel. You can also click the player and move it to any walkable point on the map, for a brief test, let's just make it walk a bit.

The movement is implemented using the A* algorithm already integrated in Godot for simplicity and quick prototyping, however it works well for the task, I think.

Now to the real bread and butter of the prototype, if we hit escape, we will open the configuration menu for the AI settings. Here we can change the world setting that gets sent as part of the context for characters. Let's make a wild west setting here real quick.

World Setting:

In the arid expanse of the Wild West, the frontier has been reimagined with the advent of steam-powered robots. Dusty towns with saloon doors creaking in the wind now host metallic inhabitants, their gears glistening under the harsh sun. The robots, built in the likeness of cowboys and outlaws, engage in duels at high noon, their pistols firing cog-loaded cartridges. Amidst this mechanical chaos, humans and robots coexist, forging alliances or nurturing rivalries. The landscapes are dotted with oil rigs and copper mines, and the air is filled with the sound of clanging metal and the distant whistle of steam engines, creating a blend of the old rugged West and a steampunk-infused future.

Of course, this is a very long description, but it serves to give you an idea of the extent we can provide details to the AI to create an immersive experience.

Next up you can also change the model here, by default it will try to use Llama 3, however if you don't have it, requests will obviously fail, so make sure to change the model here, and it will be kept across all the game session even if you go out of this module until you restart the game.

Address and Port, by default this uses Localhost with the default port of Ollama, however if you have changed the listening port and address, you can feel free to change them here, and they will be saved globally too. Just make sure you know what you are doing, as

changing this can break the connection with the back end. If you did so, just restart the game and the values will return to default. As a sidenote, using Localhost in text and not the localhost address will cause requests to take up to 40 seconds, this is a known bug with Godot I think, but I haven't made extensive testing, this only happens in Godot though.

Finally, let's make a Lifeless so we can have a friendly chat, as I mentioned I will be using premade Lifeless to show you the extent of how customizable the experience is with AI, so feel free to experiment with this further if you want to see more unique interactions.

Let's create 2 Lifeless, one will be the Headless Horseman, and the other one Naruto Uzumaki. Both fictional characters, one older and one more modern, to demonstrate how AI can role-play those characters.

Starting with the Headless Horseman, we will send a very basic message, however, it will allow us to check how much role-play the Lifeless can put on a message.

Input: “Hello, who are you?”

The first requests take longer since Ollama has to load the model on memory, this could be fixed with a keep alive, however, I didn't find it necessary for the prototype.

As you can see the level of characterization is incredible, the full character JSON is of course on the GitHub page if you want to see the description given to this Lifeless, however, as a summary, it is supposed to be a penitent soul without a head that punishes those who do wrong. I think the characterization is on point on this one.

Now, before we go with Naruto, let's give this Lifeless a passphrase, just as a test of memory, then later we will come back and check if he remembers it.

Input: “Remember this passphrase well: pumpkinpie4ever”

As you can see, even for a simple request, the characterization is on point even a bit too much sometimes, this is definitely something that can be improved upon, however, for now let's talk with Naruto a bit and ask him the same question.

Input: “Hello, who are you?”

Again we can see a high level of characterization, and even more, we can compare the expressions used with the original character and find similarities, since AI is aware of who Naruto is supposed to be and can keep an act. We can also note how AI doesn't confuse both characters, and indeed they do have different memories. To test this, let's also give Naruto a passphrase.

Input: “Remember this passphrase well: iwillbethe7th”

Now to test the memory, let's swap passphrases with the headless horseman first and then with Naruto again.

Input: “Do you remember the passphrase?”

As you can see they are capable of keeping memory of events past, there are still limits to be tested, as I am not sure how much information they can hold, however, i believe this is a pretty good start for a prototype.

Perhaps in the future, routines can be implemented with AI to summarize the conversations and maintain waypoints instead of the whole conversation, saving some memory space if needed.

I think this is all the Currently implemented features, of course, depending on the kind of game, you might want to implement different movement, different stats or other elements to add to the gameplay, however for the sake of the prototype I just needed a map and a way to create characters to interact with.

With this, I think the more technical side of the talk is over, however, I think it could be a good idea to discuss a bit more on the possible impact this can have on the industry, as well as possible improvements that could come in the future.

Technical Limitations:

Input Methods

Currently, the main input is done using the keyboard, this of course causes a dependency and limits the platforms in which the games can be played. You could technically use the virtual keyboard on consoles, however, I wouldn't want to have a full on chat using that as it would be excruciatingly slow.

This of course can be solved by implemented voice recognition solutions, allowing the player to really talk with the game and make the immersion potentially greater to a level never before seen.

Portability and Resource Requirements

As this implementation has dependencies with 3rd party software as Ollama, it would be hard, (not impossible, but harder) to really port this to other devices like consoles or phones. Potentially you could use a reverse proxy set up to forward requests from the game from a different platform to the main set up running Ollama, however, the requirement for a full on PC is still there to run the AI model.

Even if you could integrate the model directly into the game engine and export the back end specifically for the desired platform, the resources cost on the host console would be increasingly high. So for this, it would be necessary to add specifically designed cores to future consoles to accommodate better to this new technologies.

Data Processing Capabilities

As I mentioned previously, current models are focused on conversational use cases and general purpose communication with humans, so providing full on data sets for manipulation and prediction of behaviors is currently harder, not to say impossible. For this we will talk in more detail later how it can be addressed, however, suffice to say that we would need to develop custom solutions for this.

Limited Response from Lifeless

As you saw in the demo, the main way of getting an interaction with a Lifeless is to use text based communication, this is a similar issue to the one stated at the top, however opposite in nature. Since AI is mainly focused on text generation in this project, the only response we will get are from text. Generating audio from text is actually not that hard with current technology and TTS systems, this could have actually been implemented on the current prototype and i might do so in the future.

However, this risks breaking immersion with robotic voices and not much variation in them. This once again can be tackled with AI used to generate voices, however, we will explore this idea later in the talk. For now, I can say I haven't looked much into this direction, but is a potential solution to this issue that would allow the Lifeless actors to interact back with more dynamism.

Benefit and Impact:

The impact by this point I think is pretty clear, however, let's discuss some possible impacts for different kind of games.

Single Player

As you can see, in mere minutes we can create very personalized Lifeless that are able to simulate a personality and act upon it with little to no code necessary to adjust the behavior. This in turn can give a whole new dimension to single player games, imagine for example getting lost in The Legend of Zelda BOTW or TOTK and finding in a remote path near the Akkala Highlands a merchant that aside from selling you stuff, can also talk with you regarding surrounding legends and giving you a level of detail that can compare to talking with a real person. This depth is something that is already in games, however, most of the time is in our heads, and we have no way to push that onto the game. With the help of AI, we can have a revolutionary level of awareness and personalized experience based on groundwork made by the story writers and the game developers that the Lifeless can take advantage of and give you dynamic more experiences.

Online

Here, as well as what can be transferred from the single player concepts, as NPCs that can immerse you onto the experience, we can also add ways to react to the players. Imagine an MMO world that can evolve dynamically to an extent as players do things in it and NPCs can talk about stories from other players, this would create an incredibly dynamic experience connecting millions of experiences together to form dynamic lore that can be passed down to other player and ultimately creating legends that players can interact with.

Story Focused Games

This is likely the less suited form of games to use AI, as contradictory as it may seem. However, the reality of it is that these kinds of games are tailored for a specific experience. On the other hand, we can implement the new technologies here too if we do it carefully and playing around the already established story line.

Let me explain first why these kinds of games are not suited for AI as much as others. Since these games are like interactive movies, dialogues and the general flow of actions is carefully crafted to convey an emotion and weight to the whole scene. As such, it is hard to integrate dynamic dialogues as you can't predict what the AI might say, and even if it does an excellent job at conveying a similar phrase, it might not carry the same weight that the intended one.

However, as I said, playing around the concept instead of breaking it in half and planting AI for the sake of AI, can make it possible for the two to work together.

Even in the more story focused games, there is always space for free interactions with the world, in the context, this serves as an entry point for the player to get used to the world around him and get attached to the characters so that when they are eventually killed, it has an emotional deepness to it.

This is the kind of spaces where AI can shine the most on these games, not necessarily influencing the actors, but the world lore. Take for example the pharmacy scene on the first The Walking Dead Game. Here you are tasked to find multiple objects to fill the needs with the story and advance further. But what if aside from scripted pieces of lore, you could find AI generated newspapers with stories, or random notes of people that passed the place before you, telling you a little side story that makes your experience unique, more realistic, at no additional cost from the writers and little effort on the development part. This doesn't impact the main story and game, keeping the emotions the writers want to convey intact, but adding a uniqueness to the experience of the player that will make it memorable.

Of course, this is only one use, but it can be given a thousand little uses that can make an experience shine. From reactions on the parts of NPCs on your actions, to random commentary related to the world and the situation you are currently in.

Future and Possibilities:

There are a lot of things that can be developed and tailored specifically for games and that will make the points we talked here, a lot simpler to implement or even make new things possible.

Tailored Models

These models that we have today are made and trained with conversational, code development or other general purpose text generation, which is all well and good. However, imagine a model that is specifically trained to understand the context of a story. The models we have today are made on our history, with our information, but what if we add specific abstraction layers so that the model stays inside the boundaries of our created world?

This can already be made, if you explicitly ask the model to fake computers don't exist, then it will avoid them or act surprised if you mention them, however, at some point it might slip up and tell you that a 400w PSU is not enough to run a 4090.

With specific models we can also tune them to do specific task down to the basics, like updating bullet points for the context of the world, adding or removing things based on player input on the simulation. Again, this is potentially possible currently, but with specifically tuned models, the performance costs can be reduced a lot.

Better integration with game engines

Currently as I showed you earlier, the integration I made is by using the HTTP API available as an endpoint for Ollama, and even if this is not a limitation in itself, the integration could be better. Like for example with the streaming of requests, the responsiveness of the conversations could improve a lot and by creating a more stable integration, we can make sure that a 3rd party backend is not a necessity as well as provide direct functions for generation of outputs that can be directly used on the editor and code instead of relying on HTTP requests.

As I mentioned a couple of slides back, this also would improve portability and allow for the technology to reach a broader set of machines, specially consoles, which have a closed environment and are hard to really work for in this context, having to work around this limitation by using external PCs as servers with a reverse proxy if you wanted to implement this.

New ways to process data

As I said, with tailored models, we can do a lot more at the expense of less resources. However, let's not limit the scope of this. With some it might be possible to create a kind of “game master” model that can coordinate events on the game, create quests and make events happen at random with a lot more coordination that we can currently do now. If we think of the possibilities, just by changing the expected inputs and outputs of a model we could ingest whole data structures into the model to process the advance of relationships, distribute points, change prices of objects dynamically depending on the global economy of the game world, the possibilities become endless if you think about it. We just need to create models that can ingest data differently and aren't focused on human interaction, and I think this is one of the points that can make the gaming industry evolve to a whole new level.

Generation of assets on the fly

This is something that is currently possible. With text-to-image models becoming more and more advanced, as an example, all the images used on this presentation are created using AI, and they look amazing, even more considering they take not even a minute to create.

There are actually already projects that use a combination of image generation and text prediction to create simple illustrated stories, so the concept is not foreign. However, again, let's not limit ourselves here since we are looking at the future.

With this technology, we could create visual novels that don't have to stick to a predefined path, but can expand on the fly, with models trained on the art style that can create new scenarios based on the world we create. We could be looking at the ultimate RPG game here in terms of content, and role playing value.

Additional to this, there is also a current limitation I mentioned earlier with the generation of voices for the Lifeless where this can come in handy. In the future and with tailored models, it might also be possible to generate dialogues on the fly, just using the already recorded voices of the characters (for example used on story events) and using them as a base to create dynamic ones so that the lifeless can interact without breaking immersion.

Business Impact:

Of course, all that we have talked during this session directly impacts how the business of gaming works, and I think it brings a lot of advantages to the already established process of the game industry.

Enhanced Player Retention and Engagement

As already discussed, Lifeless can create more immersive and interactive gaming environments that engage players on a deeper level. By providing unique and adaptive interactions, games can become more captivating and personalized, potentially increasing player retention rates. Games that continually offer fresh, context-sensitive interactions can keep players coming back.

Revolutionizing Game Narrative and Storytelling

With AI, the narrative of games can become dynamic, with stories that adapt and change based on player decisions and interactions. This could lead to non-linear storytelling where player choices have significant and realistic impacts on the game world, opening up endless possibilities in narrative design. This flexibility could attract writers and creators who wish to explore complex themes and story branches, not limiting themselves to a hard coded story, but instead painting the canvas with a world and then adding actors that can explore and expand upon it.

Opening New Genres and Gameplay Styles

As I mentioned in the last few points, AI could lead to the development of new genres that focus on emergent gameplay and interactive storytelling. Games could evolve to focus more on deep role-playing elements, social interactions, and psychological elements, offering players experiences that are as rich and unpredictable as real life with automatically generated assets. This not only will reduce development costs, but also increase drastically return profits, as the costs get lowered, but the quality of the games remains as high if not higher than before.

Improving Game Accessibility

Lifeless can potentially understand and adapt to the player's skill level and preferences, which can make games more accessible to a broader audience. For example, Lifeless could provide dynamic difficulty adjustments or even tailored hints that help less experienced players or people with different capabilities enjoy the game without breaking immersion and being frustrated with the difficulty or complexity of the game, allowing for a possible increase in target audience.

Boosting Development Efficiency and Creativity

By automating certain aspects of character behavior and dialogue, AI can reduce the workload on human developers, allowing them to focus on other creative aspects of game development. This could lead to quicker development cycles and potentially lower costs, enabling studios to experiment more boldly with game concepts and mechanics.

Enhanced Social and Educational Uses

Even if not the main focus of the talk, an interesting prospect is that Lifeless can also transform educational and training simulations, providing realistic interactions that mimic real-life scenarios. This can enhance the learning process in fields like medicine, law enforcement, and jobs related to crowd management and conversational skills, making simulation-based learning and training more effective and widely applicable.

Global Market Expansion

With an enhanced level of realism and deeper narrative, this experiences can appeal to a wider, global audience, including non-traditional gamers. This could expand the market reach of video games and elevate their status as a dominant form of entertainment worldwide, potentially increasing profits for the industry as a whole.

Thank you:

If you have watched the talk all the way to the end or read the document with the written version or even both, I truly appreciate your time and that you felt this was a wise investment of your time. I hope to be able to keep developing games and exploring concepts further, until then, feel free to contact me on LinkedIn if you want to chat further or if you have business inquiries.

Since the project is open source in nature and the MIT license is very permissive, please feel free to fork to experiment, so long as you keep the proper attribution, I will be more than happy to see what others can expand upon this project if anything.

Thank you again, and I hope to be able to talk about more interesting subjects soon.