.slide 1

Welcome back everyone. On today's lab, we are going to go through a couple of games and look at interesting problems the designers faced when coming up with their AIs. And, since this is a lab, you are going to get a chance to come up with your own solutions before I present you with what the original designers came up with.

.slide 2

Before we get into that, I would like to mention the resources I used when coming up with this lab as well as other parts of this course, as they may be useful to you. Game AI Pro is a website that contains free articles on various AI topics from professionals, including explanations and analyses of AI techniques used in various games written by people who actually worked on them. AI and Games is a YouTube channel that has a ton of videos on how AIs in various games work. These videos usually aren't too in-depth, but they do give you the general idea of how the AI works. And, last but not least, there's another YouTube channel called Game Maker's Toolkit, which has videos on various game-related topics, not just AI. I would specifically recommend you watch their video "What Makes Good AI?", which I've linked here.

.slide 3

I don't think it contains any groundbreaking insights, but it names a lot of aspects of AI that aren't immediately obvious.

Ok, with that out of the way, let's jump into the first game, which is one I hope you're all familiar with by now –

.slide 4

Pac-man. Classic game – you play for a yellow dot with a mouth and try to collect smaller dots while avoiding four ghosts. Totally relatable stuff, we've all done it at one point. Now, what interests us here are the aforementioned ghosts – how are their movements programmed?

Before we get into that, you may remember that, during today's lecture, I promised to show you what your group project specifications should look like. So, for each of the games I present here, I will show you a short specification of the AI problem that will then be discussed – this is sort of what I expect you to deliver for your chosen game – a more extensive example is provided in slides for the assignment. And then, your presentations should be something like the explanations I will deliver afterwards, only more in depth and extensive, as you should design all the AI systems for a game, not just focus on one aspect of it.

So, back to Pac-man. To keep things simple, let's forget about the fact that one can make the ghosts edible by eating one of the larger dots, in which case their movements change. With that assumption, the specification for the problem we're facing here is as follows.

.slide 5

We have a 2D map with obstacles, where the player tries to collect all the little white dots. We have four ghosts, all of which can move across the map but not through obstacles. If they touch the player's avatar,

the player loses a life. Their purpose is to provide a challenge. The question is – how should the ghosts move?

As I said, I will now give you some time to analyse this problem and try to come up with your own solution. There are two ways we can go about doing this – either we have a discussion, or everyone comes up with their own solutions. So, let's have a vote. And, if you vote for a discussion, please do so with the resolve to actually take part in it. Now, who's in favour of having a discussion? Alright then, everyone will try to come up with their own solutions. As this isn't a very complex problem, I will give you only 7 minutes to do so, for the others, you will have 10. Please write down whatever ideas you come up with and send them to me using the form I provided on Discord – this will not be graded in any way. I want you to do this because a) writing will help you clarify your thoughts and b) I will have some feedback about whether you've managed to come up with something, whether I should explain something better, etc. You are free to talk and interact as you please, or to ask me questions if something's unclear. Are there by any chance any questions before we start?... Alright then, I'm starting a timer and please begin.

Ok, time's up. So, let's look at what the actual implementation looks like.

.slide 6

The most straightforward approach would be to just have all the four ghosts chase Pac-man all the time. However, that would probably mean that they would either follow Pac-man in a line, which would be pointless, or come at him from multiple sides, which would be impossible to escape. So, the designers had to come up with something else.

The ghosts have two movement states – chase and scatter. They alternate between these with 20 seconds of chase followed by 7 seconds of scatter.

.slide 7

In scatter mode, they move towards the four corners of the map, with each ghost being assigned a specific corner, while in chase mode, they try to move towards different target tiles. Each of the four ghosts – who have names by the way, but I'm going to omit them and just refer to them using their colours – has different rules for picking their target tile.

.slide 8

The red ghost simply chases after Pac-man, so its target tile is whichever tile Pac-man is on at the moment.

.slide 9

The pink ghost targets the tile that is four tiles in front of Pac-man (in front relative to the direction he is facing). There is an exception to this – when Pac-man is facing up, the target tile is four tiles up and four tiles to the left. However, as this is due to overflow of values during the computations, I suspect this was not the original intent.

.slide 10

The cyan ghost computes the tile that is one tile in front of Pac-man – or, again one tile up and one to the left, if Pac-man is facing up – then computes the vector from that position to the position of the red ghost and rotates it by 180° around the first tile. This results in this ghost sort of working in tandem with the red ghost to catch Pac-man from two sides.

.slide 11

Finally, the orange ghost has the same target tile as the red ghost – Pac-man's current position – however, only if Pac-man is eight or more tiles away. If the two are closer-

.slide 12

His target tile is in the lower left corner of the map.

There are several things about all this that I find interesting. First of all, it seems like such a simple problem at first glance, but the solution is in fact quite nuanced. Second, there's nothing about this that would make you think 'oh yeah, that's the obvious right solution'. That's one of the lessons that I hope to impart through this lab and the group project, because, at matfyz, we are all very focused on solving problems in the one correct way, right? You get a math problem and you have to come up with just the correct steps to solve it, or you get a programming assignment and you hack away at it until you have a program that works exactly as intended in every case. But there is no such thing when it comes to AI design – or design in general, I suppose. And third, all the rules I've just described just seem so random, yet they come together to create such a good player experience. I wonder if they went through a lot of iterations when designing this and whether there was a lot of thought put into designing these specific rules, or whether they just tried random stuff until something worked.

Ok, so, that's Pac-man. Before we move on, I just want to mention that the games I prepared for today were picked because I think they all contain some interesting AI problems that are different from the ones you've encountered in the course thus far. So, there isn't any connection between them, no overarching theme. Just random interesting problems. That having been said, the next game on today's menu is called –

.slide 13

Battle of the Bulge.

This is a strategy mobile game based on a real historic event from the second World War, when the Germans ambushed the Allies in a last-ditch effort to break apart their armies and thereby isolate them and get enough of an upper hand to force the Allies to sign a peace treaty. There are two sides in the game – the Allies and the Axis – and you can play for either one. The rules are kind of complex, but we don't really need to get into that here. All that you need to know is:

.slide 14

This is a turned-based game. The branching factor is huge. Each player has some units at their disposal which they can move around. When opposing units occupy the same region, they fight. Now, the interesting thing here is this – when choosing your opponent, you can always choose from among three different generals. These generals are supposed to have distinct fighting styles. So, the question is, how do you achieve that? As promised, you will now have 10 minutes to think it over and write down your

ideas. Just one more thing before you begin – I realise that this problem is quite a bit grander in scale than the previous one and that may make you feel aversion towards working on it. But that's also part of the point, because, if you end up working on anything that has to do with game design, these are the sorts of problems you will have to face. Now you are in a safe environment, there is no judgement, no grading, no correct answers, so please, just give it a try. Also, your answers are almost definitely going to be very different from the actual solutions that I'm going to be showing here, so don't worry about trying to 'get it right'. Just write down whatever approach comes to mind. The form is on Discord again, of course. And with that, I am starting your timer.

Alright, looks like time's up, let's take a look at how the authors solved the problem.

.slide 15

The authors' AI has three components, which are inspired by different parts of the human brain, hence their names. The first layer is the Reptile Brain.

.slide 16

This part applies various weighted utility functions to moves to determine how good they are. This will be important and we'll get back to it in a moment.

.slide 17

The Neocortex uses the move ordering provided by the Reptile Brain – as well as other optimizations – inside an implementation of Alpha-Beta Search. This is the part that actually picks moves.

.slide 18

And the third part, called the Limbic Brain, serves to learn from the player. It does this by storing what moves the player makes and when, though I'm not sure what sort of data structure is used for this – the authors just mention they use heat maps which somehow incorporate time, which I have trouble wrapping my head around, but it's not that important. What is important is that this part of the AI learns from its opponent and uses the knowledge it gains when modelling its opponent during Alpha-Beta search.

So, that's the general overview. Now, I want to get back to the Reptile Brain, because that's wherein lies the solution to the problem of personalities that I posed at the start. And I would like to mention a quote from the authors here, which I find quite insightful.

.slide 19

"Many developers try to create the perfect player first, then down tune to create personality. This prioritization discounts the great wealth of expression players have on each of their chosen battlefields. Within every game, there are very different ways to play."

So, how did the authors go about taking this into account?

.slide 18, 17

.slide 16

Well, as I mentioned, the Reptile Brain is equipped with various utility functions for moves – the authors don't go into detail about this, but I assume these can be things like 'this move brings us closer to a desirable position' or 'this move destroys some enemy units'. Generally simple stuff that can be directly constructed from the game rules. These utilities are computed and then combined using weights to give a final move ordering. And herein lies the magic –

.slide 17-19

.slide 20

By manipulating the weights, one can create different play styles. And this can be specifically done in such a way as to create some archetypical behaviour, such as 'cowardly', 'defensive' or 'rash'. Simple, but elegant, right?

And an interesting aspect of this whole design is that each of the three 'brains' can actually be built and tested independently and then put together, which, I suppose, must make it quite pleasant to work with.

The authors used this system in two more similar games besides Battle of the Bulge and I would like to close this part with one more quote from them.

.slide 21

"Each game grew in complexity, and the number of game states with it. While not explicitly a general game player, the triune brain system has the flexibility to deal with each new situation without major architecture reworking. By focusing on personality rather than winning, and data collection over code crunching, interesting AI opponents are created even in the face of complex game systems."

Alright, so that's Battle of the Bulge. For our third title, we once again have something that is pretty well-known.

.slide 22

The zombie apocalypse cooperative FPS Left 4 Dead. In this game, you play as one of four survivors of a zombie apocalypse who are trying to survive a little longer. As you try to make it from one safe place to the next, you are assaulted by hordes of zombies of the totally mindless variety.

This game obviously has quite a few AI agents – the zombies, for one thing, although they probably aren't among the most complex creations, but potentially also the other survivors. However, what we are going to be talking about today is something else – its director AI.

So, the problem is as follows.

.slide 23

You have a first-person shooter. The enemies are mindless zombies that have to come up to a player to attack them, plus some special enemies with various abilities. The players' goal is to get from one place to another. The task is – spawn the enemies in such a way as to make this challenging, while also making sure to not overwhelm the players and give them time to breathe. Please use the appropriate google form and, if there aren't any questions… Then, as before, you've got 10 minutes. Starting now.

Time's up, let's take a look at how the authors tackled this problem.

.slide 24

They created a unique agent, called the director, which is in charge of spawning and despawning enemies. It works in four phases – Build up, Sustain Peak, Peak Fade and Relax.

.slide 25

In the first phase, the director spawns a population of zombies to attack you. Then, it observes the players and waits until the fighting reaches sufficient intensity.

.slide 26

Once that happens, it sustains the population of enemies for 3 to 5 seconds – this is the Sustain Peak phase. Then

.slide 27

The director goes back to spawning a minimum of enemies and waits until all the players have had a chance to calm down –

.slide 28

That's the Peak Fade phase – followed by 30 to 45 seconds of spawning only a minimum number of enemies in the Relax phase (unless the players get to the next safe room before then, in which case the Build Up phase resumes).

Now, the critical thing here is – how does the AI know when the fighting is intense and when it isn't?

.slide 29

To ascertain this, the authors created a couple of hand-crafted metrics which, together, are meant to give an idea of the current emotional intensity for a player. These are things such as the distance at which enemies are being killed, damage taken, whether the player has been incapacitated and a special case is when one of the special enemies manages to attack a player and pin them down – in that case, their "emotional intensity score" will reach maximum. This score is tracked for every player and once the maximum is reached for any one of them, the Build Up phase finishes. Once the director transitions into the Peak Fade phase, it starts decreasing the players' scores until they get sufficiently low – if it jumped straight to the Relax phase, then that would start while there still were zombies around to fight. So, once the scores for all players are sufficiently low, the Relax phase starts and then we're back to Build Up. And that's the gist.

What I like about this and why I thought it would be a good idea to include this in this lab is that I find this to be very far removed from what we normally think of when we think of game AI, right? The closest we've come to discussing something like this during this course was the squad coordinator in F.E.A.R. which assigns goals to individual agents. So that's just to point out that this is a whole different branch of game AI and interesting in its own right.

Alright, moving on to the last item on today's menu, we have another well-known zombie game –

.slide 30

The Last of Us. In case there's someone who doesn't know, The Last of Us is an RPG where the player finds themselves in a post-apocalyptic world where a fungus has infected much of the population. They must traverse the ruins of cities while defending themselves against both zombies and humans. They are accompanied on this journey by an AI companion named Ellie – or rather, storywise, they are accompanying her – and she is who we're going to be taking a look at today.

The AI companion in this game can do quite a few things, such as actively helping in combat and gifting the player ammo or health packs. Here, we are going to focus on just two foundational behaviours. So, here is the specification.

.slide 31

You have an open-world RPG. The player has to use stealth to avoid and ambush enemies. Ellie, the AI companion, constantly follows them around. The questions are – what following behaviour should the companion exhibit? How should the companion behave when the player is in stealth mode? And how should these behaviours be implemented? Once again, the corresponding google form can be found on Discord aaand, unless there are some questions… you have your ten minutes.

Time's up, let's take a look at how the authors tackled this challenge.

AI companions can be quite a bother – from walking at half the player's speed to actively getting in the player's way, there's a lot that can go wrong when implementing this. The developers of The Last of Us were aware of this and were determined to create a believable character that the players would genuinely care about – which, by the way, led them to scrapping their entire system 5 months before shipping the game and starting from scratch.

I would like to start by sharing two quotes from one of the authors.

.slide 32

"Characters give the illusion of intelligence when they are placed in well thought-out setups, are responsive to the player, play convincing animations and sounds, and behave in interesting ways. Yet all of this is easily undermined when they mindlessly run into walls or do any of the endless variety of things that plague AI characters. Not only does eliminating these glitches provide a more polished experience, but it is amazing how much intelligence is attributed to characters that simply don't do stupid things."

.slide 33

"As a general rule, characters don't need complex high-level decision-making logic in order to be believable and compelling and to give the illusion of intelligence. What they need is to appear grounded by reacting to and interacting with the world around them in believable ways."

In accordance with this line of thinking, the people behind The Last of Us put great emphasis on perfecting all the little behaviours of their AI agents, and especially Ellie, so that they would come off as believable. The most basic of these behaviours, for Ellie, was following the player.

.slide 34

To ensure that she was always in a reasonable position, the authors started by defining a follow region around the player. Then, they did three steps of raycasting. In the first step, they cast rays from the

player to the follow region to make sure there was a clear line of movement there. Each of these rays generated a candidate position, if it reached the follow region. Then, from each of these positions, they cast a ray forward to make sure that the character wasn't going to walk into a wall. This generated a set of forward positions. Finally, from each of *these* positions, the last set of rays was cast back to the player to make sure that Ellie wouldn't wind up putting an obstacle between herself and the player. Without this, she could walk behind fences or the other sides of doors.

So, that takes care of normal walking. Now, what about stealth mode?

.slide 35

To find a good cover position for Ellie, the authors started by reusing a tool they were already using for the human enemies, who also take cover when fighting with the player. This tool analysed the collision mesh and pre-calculated potential cover spots. This was good enough for enemies, but not for Ellie. The authors therefore added a system where rays would be cast from the player's position in a circle and then some analyses would be performed on their intersections with the scene to find more potential covers. Each ray generated an intersection point and the normals and locations of the points were then compared and nearby points with a similar orientation were combined into a single cover.

The two sets of potential cover places were combined and compared using various metrics to pick the best one. These metrics included current visibility to enemies, predicted future visibility, and proximity to the player.

So that's basically their solution to the problem that I outlined. However, this doesn't really do the source article justice, because it's filled with these little insights about what the designers did to make Ellie's character feel more human. For example, when implementing Ellie's follow behaviour, they had to decide what she should do when the player would try to walk over the place where she was standing. Their first thought was to just have her move out of the way so as to not impede the player's movement. However, they decided that this seemed unnatural and, in playtests, they found that players usually didn't try to run into her if she was standing somewhere. They therefore decided that she wouldn't get out of the way until the last second and then would reprimand the player for their behaviour.

Or, another decision they made was to almost never have Ellie cheat in any way, such as teleporting or dealing more or less damage than the player, as that would make her feel less natural and break the immersion.

And there are many more of these nuggets of insight into other behaviours in the article about Ellie and then also the articles about human enemies and infected enemies, so I definitely recommend you check them out if you're interested in these sorts of things.

And, with that, we've reached the end of today's lab. However, you may notice that there is still one more form left in the list that I posted on Discord. And that is because this is the first time I've tried making a lab like this and I would like some feedback.

.slide 36

You will be asked three questions. First, were you satisfied with the information I presented today – did it seem useful to you? Second, what did you think about the parts where I gave you time to think about the problems I presented? Did you find it productive, or not really? Do you have any idea about how to

make it better, maybe? And, last but not least, in two weeks, you are scheduled to have one more lab before Christmas. If you look at our website, however, you will find that it hasn't been scheduled yet. So, there are two possibilities. If you liked this lab, we can have another one like it, where I will present four or five more games. Or we can skip the last lab and you'll be free to spend your time however you wish. Please answer everything truthfully. If this was the worst hour of your life, do tell me – so that I can fail you. Joking of course – the feedback's anonymous, so I couldn't even if I wanted to. And once you're finished, you are free to leave. See you next week.

.slide 37