DefencePresentation

# First page

Hello I am Matthieu van den Berg. For my graduation project I have made a game called casino inc.

Today I’m here to tell you about the process I went through.

# Index

In today’s presentation I’ll be talking about these different things:

A short introduction

The goals for the project

The influences that have led to this project

The milestones I used to plan.

And the end result.

# Introduction

During block C and D, I have done a personal project to graduate. In this project I have made a casino management game.

It is a Tile based top down game, where the player get’s total freedom of how to run their casino.

# Goals

My goals for this project where to make a playable game, and learn new AI systems.

Both will help me further my portfolio and make me a better programmer.

For the AI systems I want to use Goal oriented action planner as I have not used that before, but is often used in simulation games.

# Influence

As influence I have used another brick in the mall, prison architect, and academia: school simulator.

All three are quite popular tile based top down management simulation games. And in my opinion very fun games to play, having taken up many hours of my free time.

# Specifications

The specifications that I’ve used to describe this project are as follow

The theme is a top down tile-based build your own casino. Where the player get’s freedom to build in whatever way they want.

The main gameplay loop is fairly simple:

1. The player spends the money they have to build/expand their casino.
2. The casino is being build by builder AI’s in the way the player has specified in step 1.
3. Guests come to play games in the casino, making the player money to return to step 1.

The motivation buzzwords for the player are creativity and bigger is better.

Looking at the games that I’ve mentioned as influence you can see by all 3 that players are driven by making their prison, mall, or academy bigger and better in any way they can think off. Resulting in more money to get more creative in how to build the next parts of their buildings.

The building system that is used is an indirect system, in this system the player can choose what needs to be build, and where, but it doesn’t instantly appear but instead is queued for an AI builder to make it.

For the AI I will be using a combination of Goal Oriented Action Planning, or GOAP for short and utility theory.

The staff will work using only GOAP, and the guests will also use utility theory. I will be going deeper into this when we’re at their respective milestones.

To make sure I was keeping on track with everything I used the management tool HackNPlan.

This is an online multi-tool with multiple things like: Kanban board, metrics, design doc, etc.

I’ve used this a lot to keep track of the different tasks I’ve done and planned. For those tasks it both has an option to put in expected time for it, and a log where you can log the time spent with an explanation of this time.

# Milestones

For planning, I have divided my project in 10 different milestones for the features I have made:

1. 2D tile map
2. Basic player interaction
3. Pathfinding
4. Ai task manager
5. GOAP planner
6. Casino management
7. Builders
8. Poker, blackjack, and slot machines
9. Guests
10. Dealers

I’ll be telling a bit about each of these milestones.

# 2D tile map

The tilemap is the part of the game that shows everything, as the name implies it’s a map made of a grid of tiles.

I’ve used this as this is the style that the preview games use to, and it’s a staple of a lot of different other management games.

For this I use a combination of Unreals build in tilemap, and my own made tilemap.

This combination is made because the unreal tilemap is purely cosmetic, so only draws the tiles, but for this game I need separate behaviours for different type of tiles.

The system I made myself to give different tiles different behaviours consists of a list of actors with the bare minimum, which have all information about what type of tile it is, and some shared info like if its walkable and which sprite to use for it. With added on top of this are actor components for the tiles that need additional behavior.

# Basic player interaction

With the map in place, I wanted to start by adding the first controls for the game, like moving the camera around, zooming, and in game ways to edit the map.

This was all fairly straight forward, but needed to make it a game instead of a picture.

I also added a first bit of UI, to show what tile the player is hovering over, and some basic tile manipulation. As seen on the pictures.

# Pathfinding

To make the AI’s able to walk around without phasing through walls, I needed to make some kind of pathfinding.

For this I’ve used a A\* type algorithm using the tiles as nodes.

A-star is a widely used pathfinding algorithm, working with a so called “best-first search”.

It does this by making a list of nodes right beside the starting location, and from there chose the next node with the lowest combination of distance from star + heuristic distance to goal. For this heuristic score I used the straight-line distance to the goal.

The biggest problem with this in itself was that I have not done any pathfinding in a long time, but after watching a few video’s and reading through the A star Wikipedia page I got this done fairly quickly.

A bigger challenge came up when I needed to make this work in a multi-threaded way to not make the player wait for a few frames when a character needed to go to a new location. Which was a bigger difficulty as I had not done real multi-threading at all before.

However, Multi-threading this was not the worst to do, everything was already really self-contained, and mostly just needed to be moved off the main thread. The biggest problems however came with returning the found path to the main thread in a safe way. Multiple sources I found on this told me that this should be done with unreals delegates which are supposed to be thread safe. After a lot of debugging I however found that this was not the case, and the delegates would be called on the same thread as the pathfinding was calculated, so I had to find a different way for this.

After some more research I got told about the TQueue class in unreal. This queue was actually thread safe, in a way that only the main thread can take data off this queue, and you could put data onto it from as many sub-threads as you like without problems.

With these found paths, I then am able to make the AI walk through the world without problem, by following it from tile to tile.

# AI Task Manager

To keep track of which tasks the player gives to the AI’s, there’s also the need of a AI Task Manager, this consists of a actor with as main function 2 hybrid First in first out queue’s, with the addition that it will take the first task allowed for the AI in question, so not always the very first task. Example of this is when there’s a queue of building 2 walls, cleaning trash, building another wall, and working at a poker table. The returned task for a dealer will be the work at a table task and not the build wall task, because he can’t do any of the others.

Another thing that I wanted to have added to the task queue is a way to take tasks to the front of the queue to prioritize them, this was done by simply adding a 2nd queue which will be worked in a similar way, but always checked before the normal queue.

In the end a queue could look like the following:

It starts with a queue filled with one build wall task, a clean trash, two more build wall tasks, and ends with a work at poker table task.

The player decides that the clean trash and 3th wall are needed earlier than the other things, so puts a priority on them.

then the tasks are done in the following order and persons:

builders will do the build wall task in order: 3, 1, 2

a janitor will do the clean trash task

a dealer will do the work at poker table task

# GOAP

For the AI decision making I wanted to use a system I’ve not used before, which is a Goal Oriented Action Planner, or GOAP for short.

GOAP is a system where the AI plans a set of actions that will take it from its current state till a specified goal state.

For example, a builder wanting to build a wall would need building tools, and building materials.

An AI using GOAP will then be able to think out itself that it needs to pickup these tools from a tool storage, and materials from a storage room before it can go to the build location.

But if the AI already has tools, it creates a plan of action with only getting materials before going to the build task.

For this action planning I’ve used an algorithm very close to A star pathfinding, but instead of location nodes, it’s using the actions as nodes.

The start and end of the path are the states that the AI needs/has, and using the difference between this as heuristics you can create a path from action to action between the goal and start.

A full cycle of a builder trying to go and build a wall would be as follow:

His current state is empty, and he needs to get to the goal state of having tools and materials.

The planner adds the actions to the nodes to check.

It closes the “drop tools” and “drop materials” nodes because these are not doable with the current state.

it checks what happens to the state when doing any of the leftover nodes, and creates copies of the state for this.

then it creates new open nodes with the possible tasks.

In the next step it checks what the outcome of these nodes would be.

And then it takes one of the branches that reaches to the goal.

Just like the Pathfinding, this is all done on a different thread from the main one, this time I had a lot less problems with this, because I still had all the recent memories of what I had done for making the pathfinding multi-threaded.

# Casino Management

The next step is adding some functionality for keeping track of the different stats about your casino and a day night cycle.

For the day night cycle, I’ve added some visual cues to indicate its night or day, and a clock in the UI.

Other stats that I’ve added tracking for is the amount of money the player has, how much they made that in game day, the amount of staff and guests, and a happiness meter.

# Builders

All the previous milestones lead up to this one. With builders we put all the systems build previously together into one, and really start to have a game about building a casino.

The builder does the bulk of the work for your casino, as the name says it builds it all for you.

When they get hired, they enter your casino from the sides of the road, and ask the AI task manager for a task.

When they’ve got a task from the manager, they will use the GOAP system to figure out a way to get all the things they need for this task, after which they will follow this plan to end up building your casino!

# Poker, blackjack, and slot-machines

As games for the Guests to play, I wanted to make 3 different once, slot-machines, poker, and blackjack.

Each of the games has its own chance of how much a guest can win. These tables of chances are created and edited by a designer in data tables which can both be changed in excel, or in a nice editor unreal has built in.

Each instance that is being build in game only allows for a certain number of guests to use them, and when interacted with by guests, they will be played for a amount of time, after which a random weighted result is added to the stats of the guest.

# Guests

The guests are the AI’s that make you able to make money. They enter the casino at specified points in the world where all the AI’s spawn, and from there go around your casino playing games and spending money.

They choose what kind of game they want to play with the use of a very simple utility theory type decision making, each type of game has a score depending on what the guest likes, and if their stats are in a good enough mood to play. And this score will be used as a weighted chance on which game they actually decide what to do.

Adding to this score is also guest personality’s, so some guests like playing poker more, and others like to play slot machines more.

This also influences how effected the happiness of the guest is for winning or losing, for example a player that likes to play poker more is going to be happier when they win in poker, than when they win on a slot machine.

# Dealers

The final AI I have made is the dealer. Dealers are needed to let the guests play poker or blackjack, where they will give the cards to the players.

To decide which tables they go to, the dealers use the AI task manager, which has tasks for working at a table.

# Results and conclusion

This all has come together in a small game I’ve been calling Casino Inc. I am very happy with how it all ended up working and looking, and think I’ve made a very fun little game. Which works very well.

The player is able to fully build and run their own casino, and watch the guests go around it playing the games.

This also gives a good basis which can still be expanded massively, for example with the guests needing food and drinks, more games, and more types of AIs such as janitors to keep everything clean.

All with all, I am very proud of what I’ve managed to do this block, and how it all went without very big dissapointments.