HOW WE WON GAMEDEV BY ROLLING OUR OWN TECH

Mihai Gosa
Co-Founder/CTO at KillHouse Games

GAME DEVELOPERS CONFERENCE March 14–18, 2016 · Expo: March 16–18, 2016 #GDC16

UBM

LET ME INTRODUCE MYSELF…

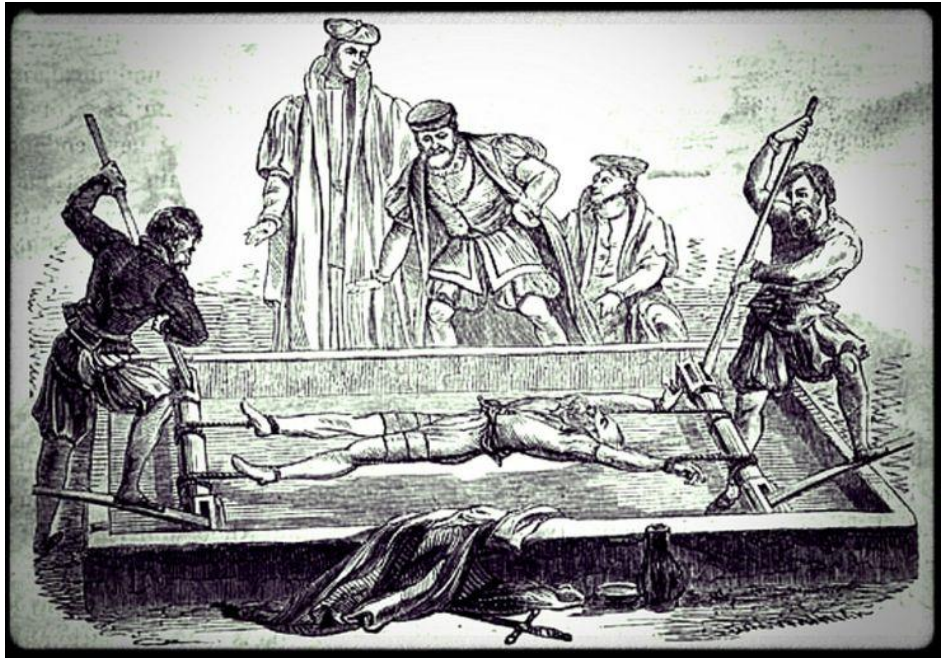
- Ex Ubisoft, Ex EA
- Now doing a bit of everything @ indie studio KillHouse Games
- Released ‘Door Kickers’ in 2014, now working on sequel.

WHAT'S THIS ABOUT?

What if I told you…there is value in NOT using a 3rd party engine?

You can make games **faster**
You can make games **cheaper**
You can make games **better**
…by building your own tech?

HERESY YOU SAY!



HOW?

Bird in space (by C.Brancusi) – $28M

… now let’s apply this to game tech

SIMPLIFY!

"Simpler" is:

- faster
- cheaper
- more efficient
- less buggy

So, how "basic" can you get? Let's see…

DOORKICKERS


WELCOME

DOOR KICKERS

- 2014's best tactics game (acc. to "Rock, Paper Shotgun")
- 97% ‘overwhelmingly positive’ rating on Steam
- Over 300k copies sold on 5 platforms
- Grossed $2M+

DEVELOPMENT EFFICIENCY - CONCEPT

Remember the 90s:

- Games much more complex than some of today's would only take a fraction of the time to develop
- Where did we go wrong?

So, can we:

- Create successful games without taking years on development?
- Without spending big money on tools?

PROOF OF CONCEPT – DOOR KICKERS

Very low cost + fast delivery

- Only 1 programmer initially
- Custom-made engine developed in 3 months
- Budget of $1500 up to Early Access version
- Available on 5 platforms (~2 days coding for each implementation)

PROOF OF CONCEPT – DOOR KICKERS

Huge amount of content:

- Over 120 maps modeled after realistic building plans
- Random mission generator
- Five campaigns
- Modding support

STARTING GROUND: WHY ROLL YOUR OWN TECH?

Limited budget (we had 6 months to make it or break it)

- Unity/UDK can take from 1 to 4 months to use proficiently (without previous experience)

Using an engine raises the expectations = more dev effort

- If you have support for real-time GI, Flash UI, you may feel the urge to use them

Learn something in the process

- Programmers also need to have fun / be creative)

You’re cheap and don’t want to give % of your game away

PROOF OF CONCEPT - HOW?

Can we make something

- so **simple**,
- so **efficient**,
- so **streamlined**,
- so **low-cost** and **fast**

that it outweighs the benefits of using a third-party engine?

BLASPHEMY YOU SAY!

PROOF OF CONCEPT - HOW?

Get close to 100% efficiency in development

- Set fixed, final goals early on: only develop what you need, not what you "might" need
- Embrace limitations and restrictions (design & tech): do you **really** need that cool animated GUI?
- Ask not what you can add, but what you can remove

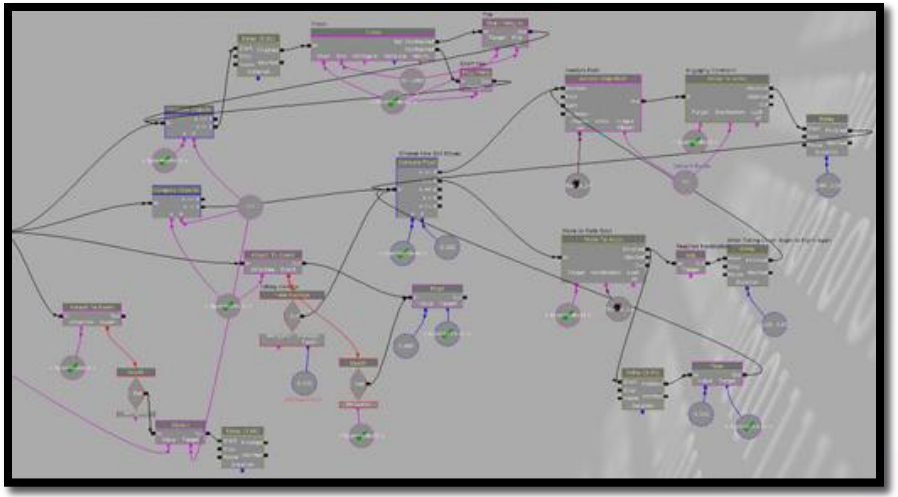
AN EXERCISE IN REMOVING…

Stand alone editor?

- *In-game editor: faster to develop, direct pipeline, WYSIWYG*

AN EXERCISE IN REMOVING…

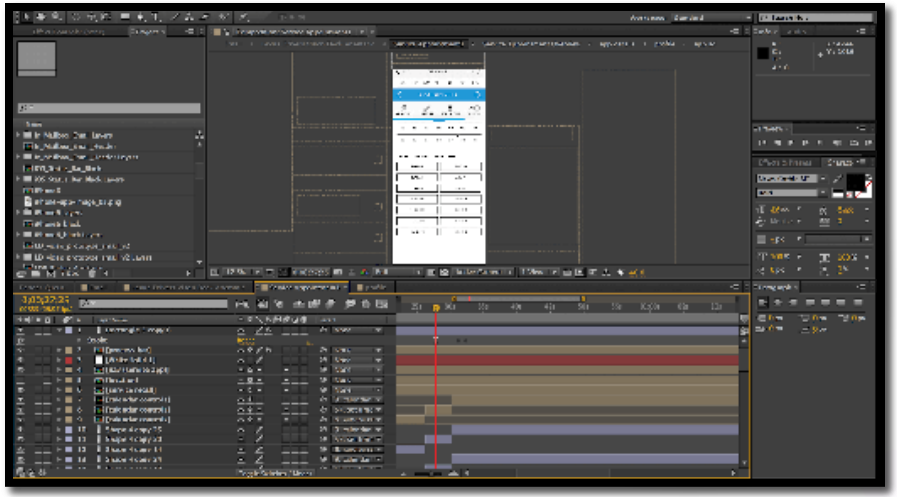
AI scripting per map?



- *Should place AIs in a map and they should handle themselves.*

AN EXERCISE IN REMOVING…

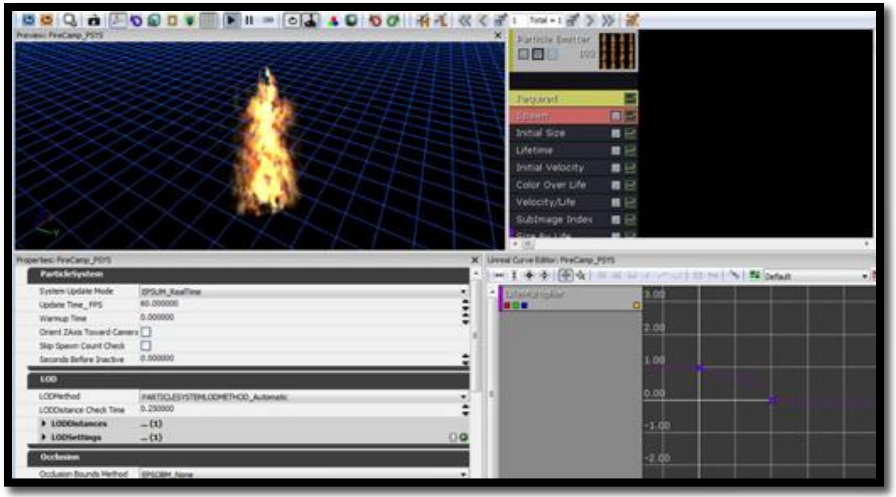
UI Editor?



- *Who doesn't prefer fast/simple UIs? Don't you hate those games with low FPS in the main menu?*

AN EXERCISE IN REMOVING…

Particle system?



- *Why not make sprite animations for everything?*

AN EXERCISE IN REMOVING…

2D animation system?

- Too much work!

Specifying collisions, lights and entity properties?

- Too … much … work!

WHAT WE NEED...

PUT IMAGES ON SCREEN AND IT SHOULD ***JUST WORK!***

WORKFLOW

1. Artist makes PNG images

2. Designer adds PNGs to the map (in a very professional way!)

3. Run game: everything works

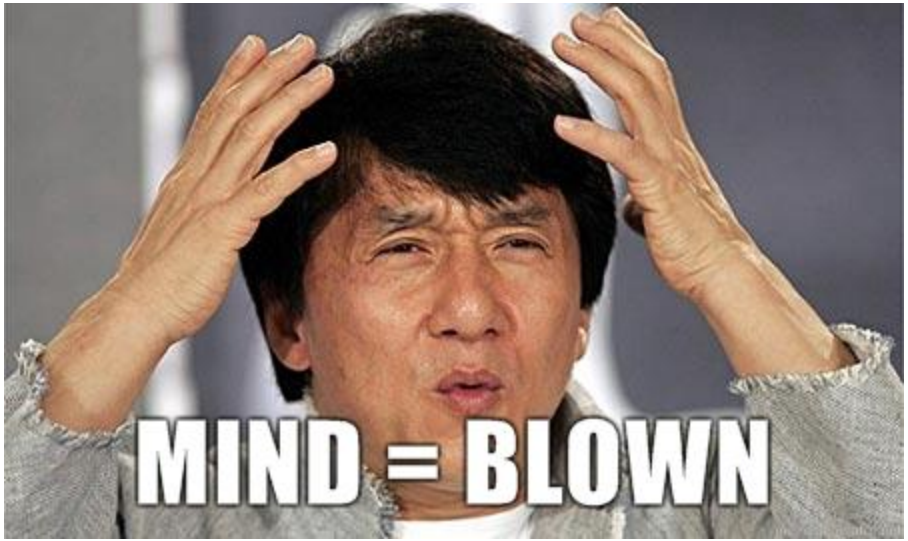
THIS IS WHAT YOU'RE PROBABLY THINKING BY NOW…

"Engine" →

WELL, WE ACTUALLY DEVELOPED THE BEST GAME ENGINE IN THE WORLD

Unified **everything**

UNIFIED **EVERYTHING**



WELL, WE ACTUALLY DEVELOPED THE BEST GAME ENGINE IN THE WORLD

Why?

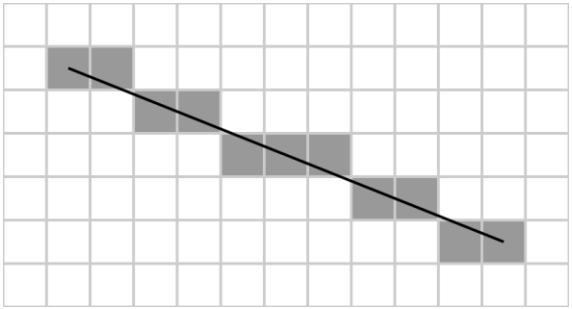
- Unified **everything**
- Cheap: no paid third-party software
- Very few bugs, very fast
- Huge return on investment

UNIFYING EVERYTHING

Can we find the lowest common denominator for all system in a 2D game?

- Could all game data be contained in a 2D image?
- Could an image-based approach be used for all systems?

Bresenham algorithm ftw!



UNIFYING EVERYTHING

At runtime:

- Render all entities in a single RGBA buffer
- Generate collision map
- Generate pathfinding map
- Generate direct lighting/ambient occlusion

No pre-processing needed!

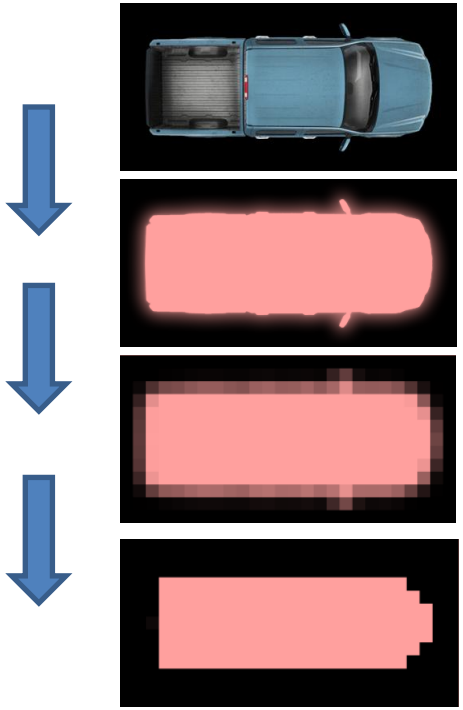
COLLISION MODEL GENERATION

Render images to screen

Entity ID instead of color

Downscale 8x

Apply threshold to remove shadow



COLLISION DETECTION

8x downscale

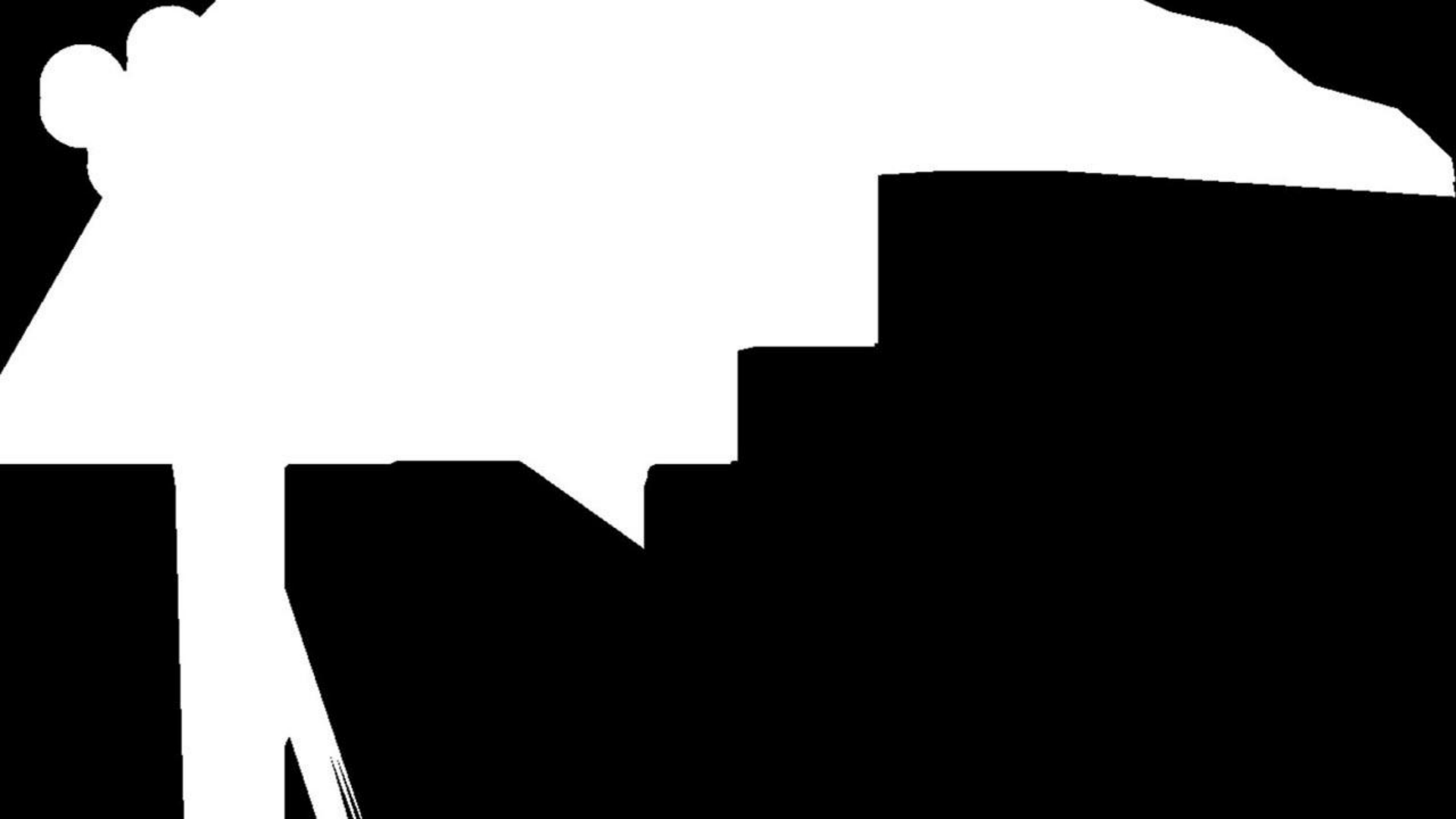
PATHFINDING

A* on **the same collision map**

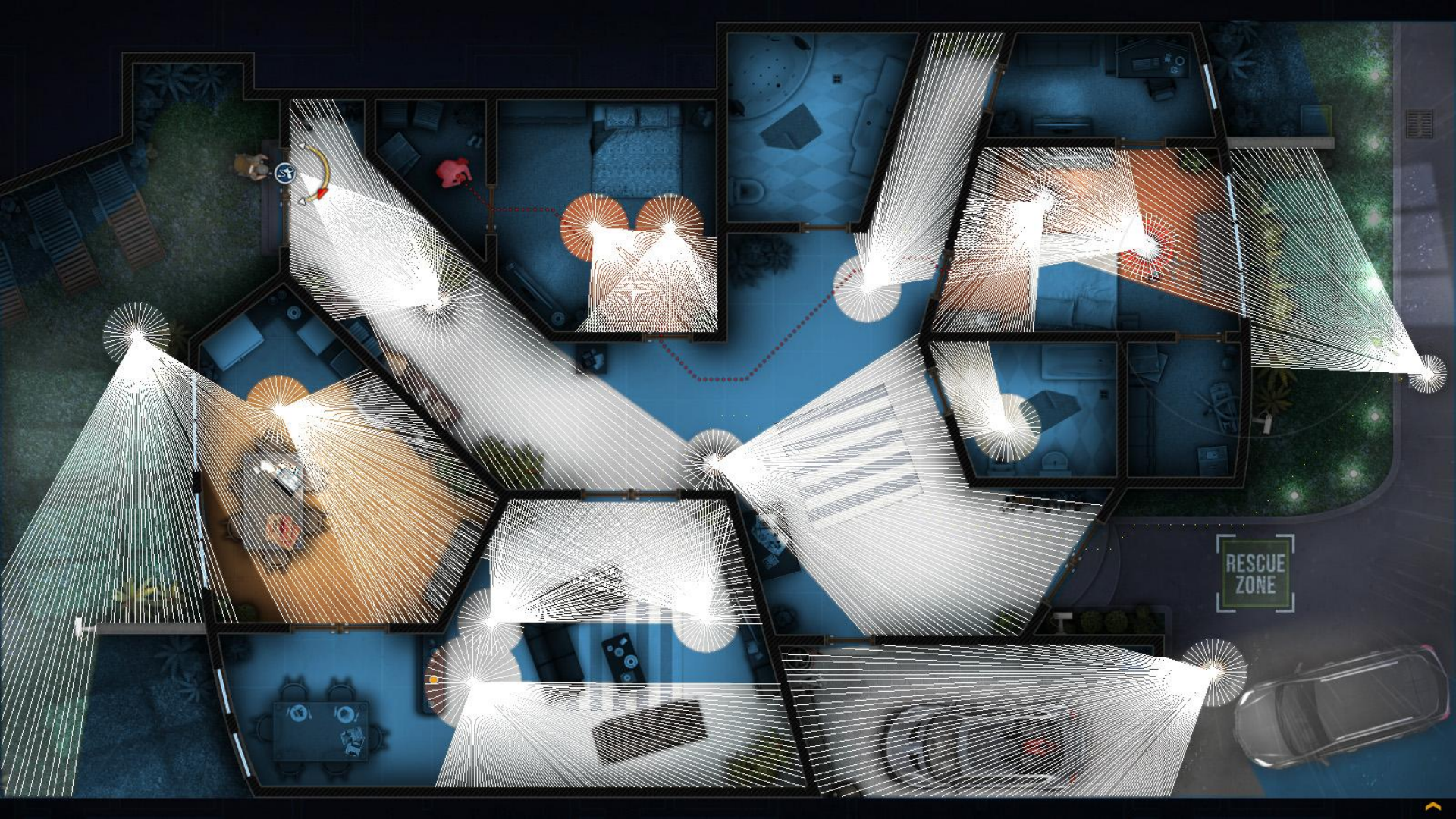
FIELD OF VIEW

- Shoot 360 rays for each character, **in the same collision map**


RESCUE
ZO

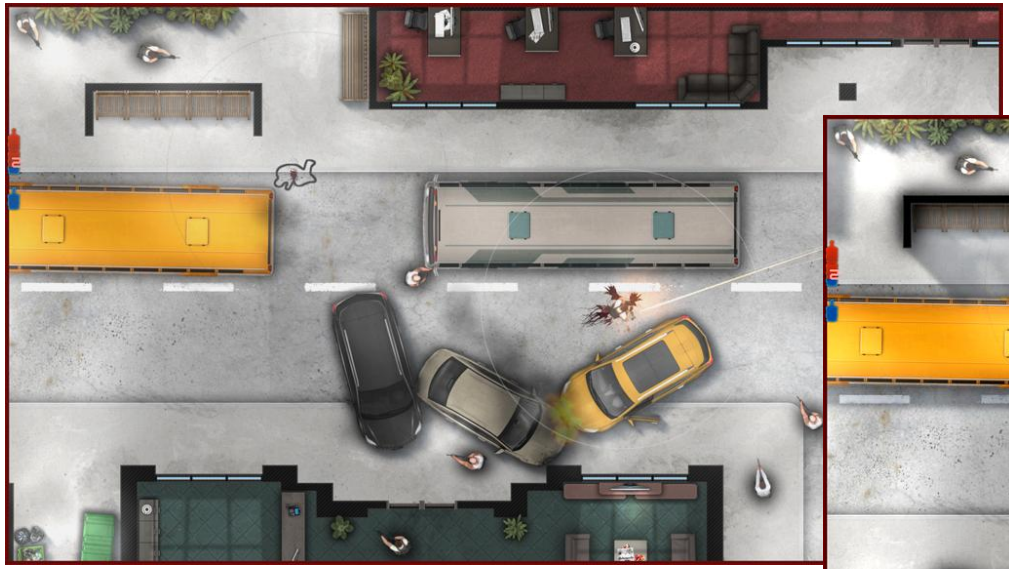



RESCUE
ZONE


RESCUE
ZONE


RESCUE
ZONE
POLICE
896
823

DIRECT LIGHTING & AO



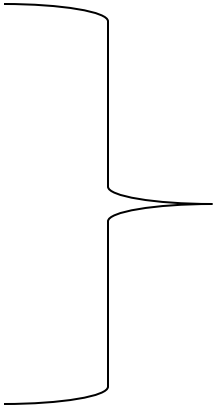
AO and directional

DIRECT LIGHTING & AO

- Ray-trace using Bresenham in the same collision map
- Only done once, fast enough for load-time

ONE IMAGE TO RULE THEM ALL

- Bresenham for:
  - collision detection
  - field of view
  - lighting & AO
  - sound reverb
- A* for pathfinding



Same dataset (image) for everything!

NOT SIMPLE ENOUGH?

How many file formats do we need?

- Game is basically an XML, TGA and WAV loader

Least common multiple for graphics API?

- OpenGL ES 2.0 (use OpenGL subset on PC, native on iOS/Android)

STILL TOO COMPLICATED?

Premature optimization is the root of all evil

- Why bother with texture atlases?
- Zipping game resources would just make modding harder
- Less than 1GB of data? Don’t bother optimizing for disk space or RAM
- 2D/sprite animation system? Just make all frames equal-sized quads.

BUT IMPLEMENTING MULTIPLE PLATFORMS IS HARD...

This is our operating system layer for Win/OSX/Linux/iOS/Android:

```
int             OS_CreateWindow(int width, int height, bool bFullScreen);
void            OS_DestroyWindow();
void            OS_RetrieveSupportedDisplayModes(...);
int             OS_SwapBuffers();

const char*     OS_GetWritableGameFolder();
bool            OS_CreateFolder(const char* szPath);

unsigned int    OS_GetTimeMsec();
void            OS_GetLocalTime(sSystemTime& time);
void            OS_ProcessInput(InputQueue& inputQueue);
```

BUT IMPLEMENTING MULTIPLE PLATFORMS IS HARD...

OSX/Linux implementations took less than installing the actual operating systems

iOS/Android trickier due to the myriad of aspect ratios and resolutions

Currently considering PSVita and WiiU ports

DISCLAIMER

No two games are alike, but challenging the status quo can be done everywhere

- We're currently applying the same approach on the fully 3D sequel
- Probably won't work for a finely-tuned AAA game ☺

This approach only works when developing from scratch

- Don't bother if you have a well-established framework in place

END / QUESTIONS

Email pintea@inthekillhouse.com
Twitter @gosamihai

QNGZ0-M8Z7A-8LXMV

B8TLA-5W5TW-I9WN3

6CPVE-TYL77-AZ47N

PIBM6-0JG65-Z2HA5