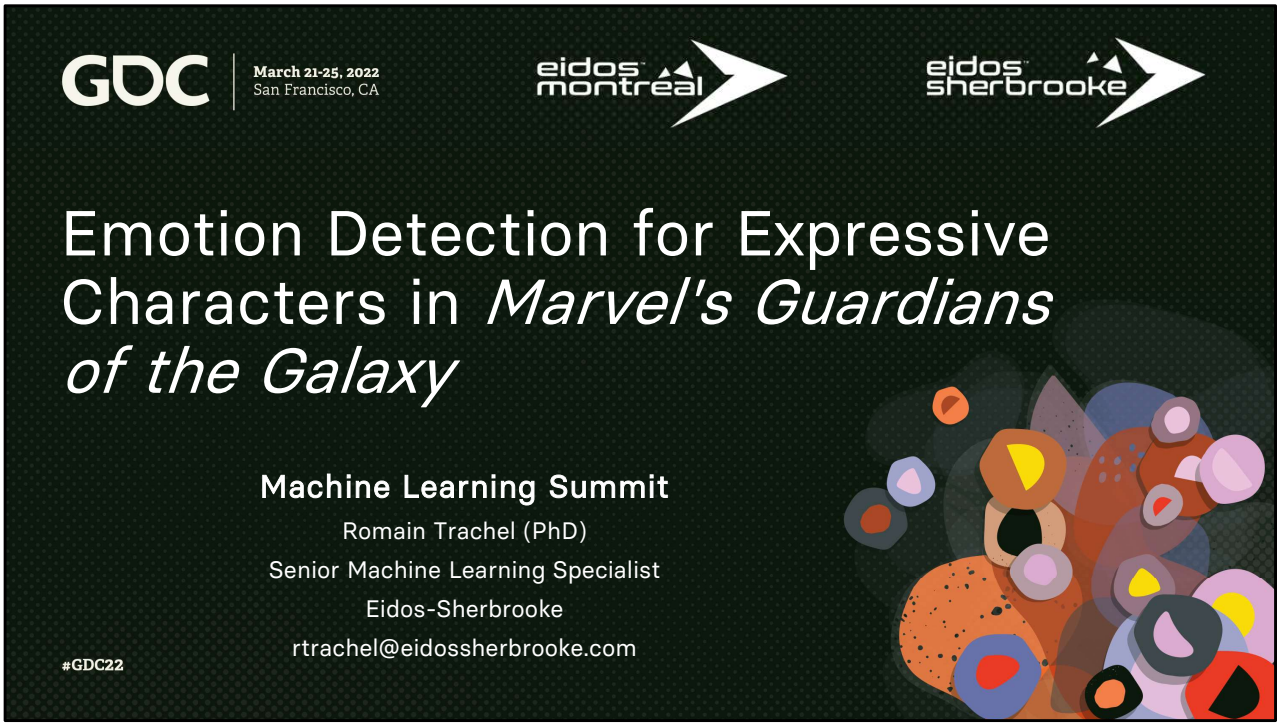
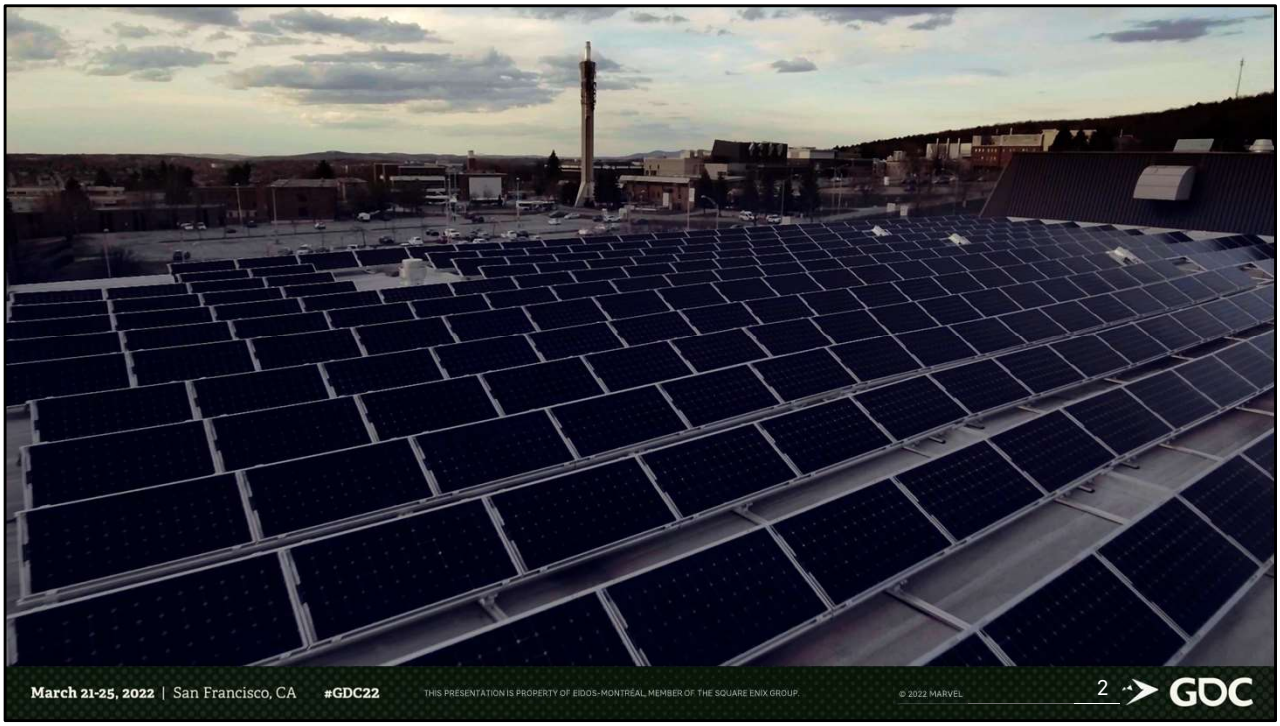
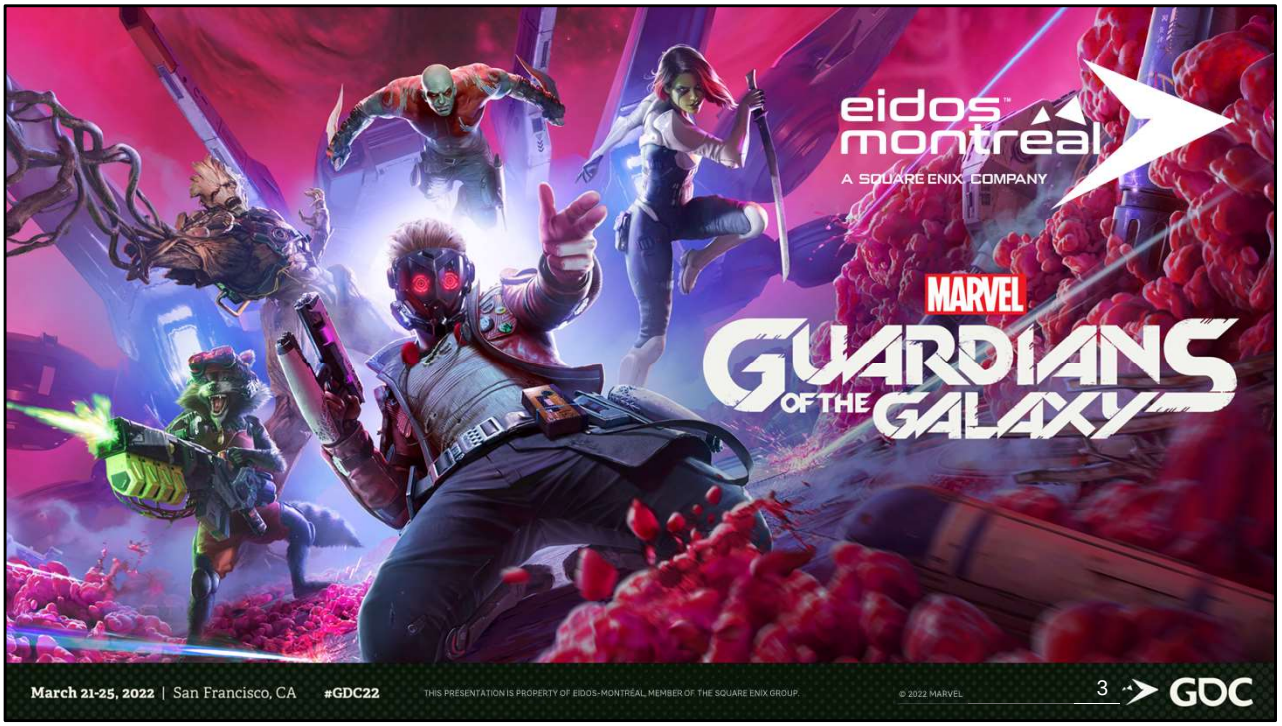


Hello, thank you for watching my presentation. My name is Romain Trachel, I am a Senior Machine Learning specialist working at Eidos Sherbrooke. This is my first time at GDC, I am very excited and honored to present my work about Emotion Detection for expressive characters in Marvel's Guardians of the Galaxy. It was a collaboration between LABS a research team at Eidos Sherbrooke and a production team at Eidos Montreal. Eidos Sherbrooke is a sister studio of Eidos Montreal, it was created 2 years ago for video games innovation.

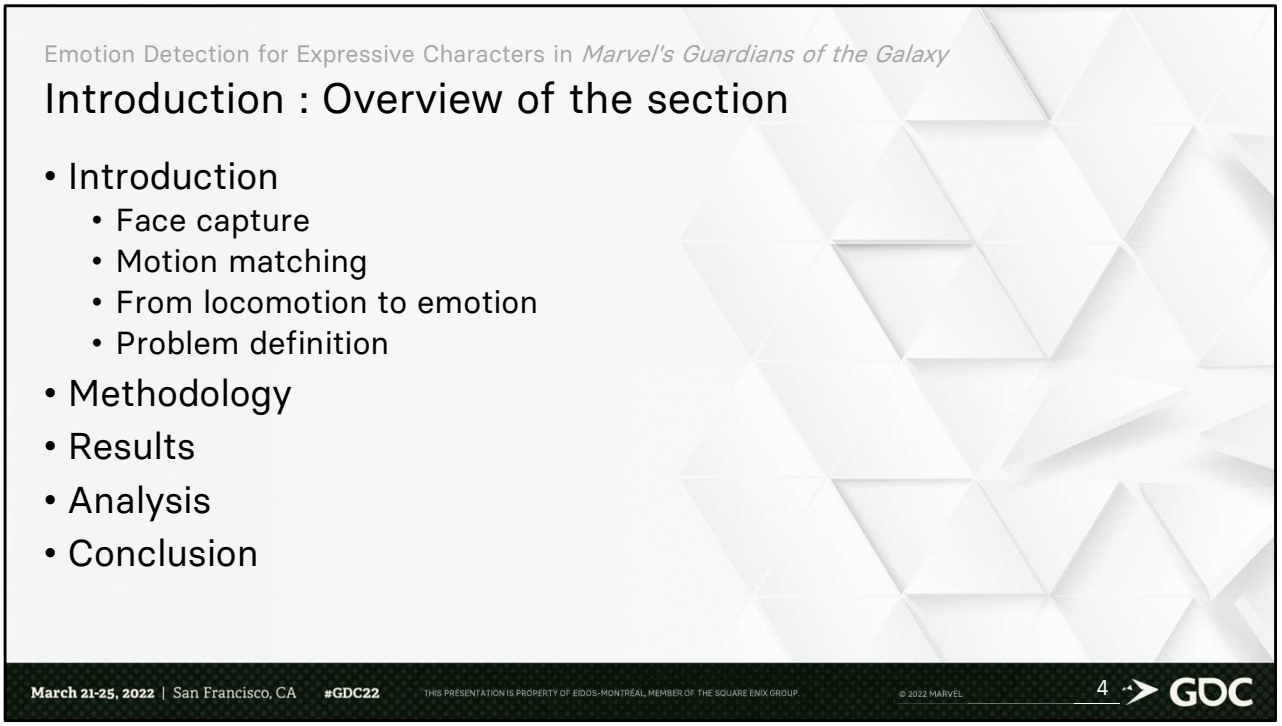


Here is a brief presentation video about the sherbrooke studio.

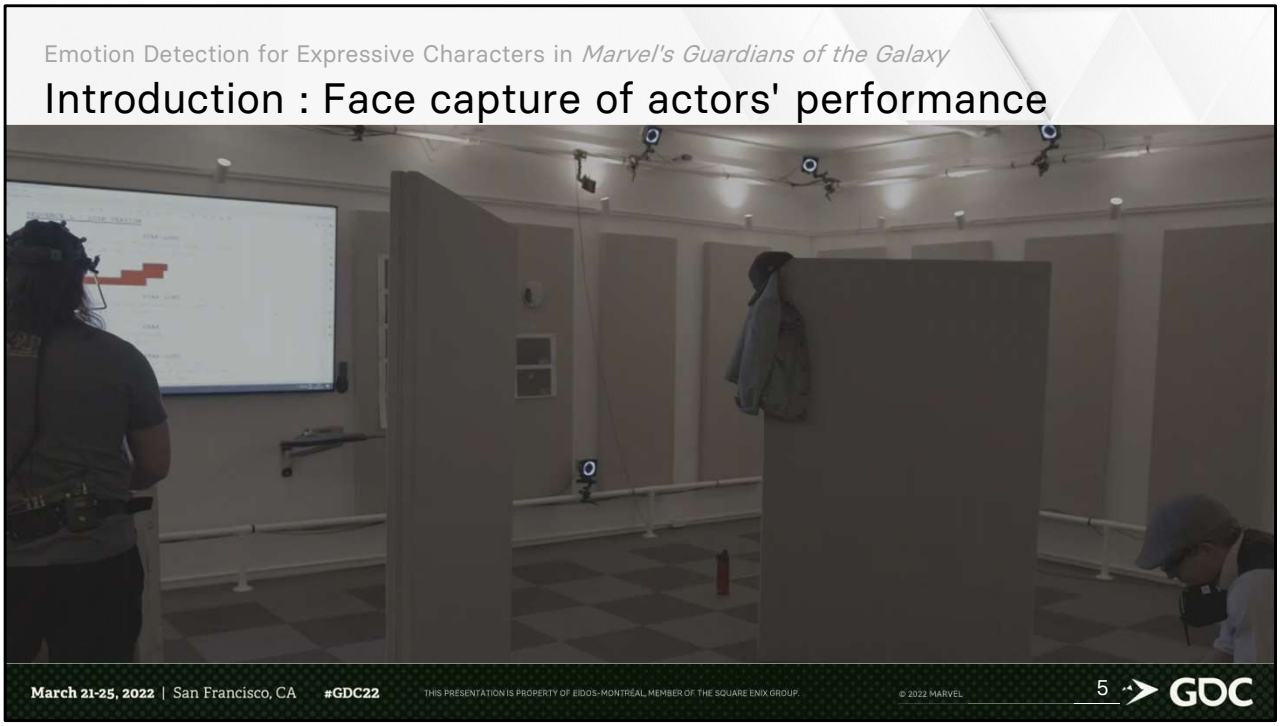


Marvel's Guardians of the Galaxy is an award-winning, story driven, adventure game that was released last year by Eidos Montreal. There's a rich narrative content between the player character (Star-Lord, the leader of the guardians) and his 4 teammates (Gamora the deadliest woman in the galaxy, Drax the Destroyer, Groot and Rocket the Racoon). It was challenging to make expressive characters such that, while playing the game, the player could feel like he or she is actually "hanging out with the Guardians". The animation staff had to handle a lot of content to avoid repetition and make these characters come alive. But they couldn't be

scripting every animation sequence manually, they had to be smart and make it within the production budget. That’s how I contributed to the game development, I helped them to ship the game with a new animation technique powered by machine learning.



I've split my presentation in 5 sections, there's quite a lot of content, so I will try to speak slowly and leave time for questions. Here is the overview of the introduction where I'm going to present the animation technique used in production. It will help to better understand the problem that machine learning is solving, and why it helped the animation staff. Then, I'll describe the methodology used to deploy my solution in production, I'll show some results and the kind of analysis I did I during this project. Finally, I do a brief post-mortem and I'll share a few key takeaways with you at the end of the presentation.



Like in many AAA games, characters animations were recorded in a mocap room with full body and face tracking. The actors' performance were recorded live, here we can see Star-Lord and Drax, reading their scripts with a head mounted camera for face capture [START VIDEO]. This kind of data was recorded for Facial Animation, and I used it to detect emotions from actors' performance. If you'd like to know more about the Facial Animation pipeline, there was a presentation yesterday by Simon Habib in the Animation summit.

Introduction : Motion matching

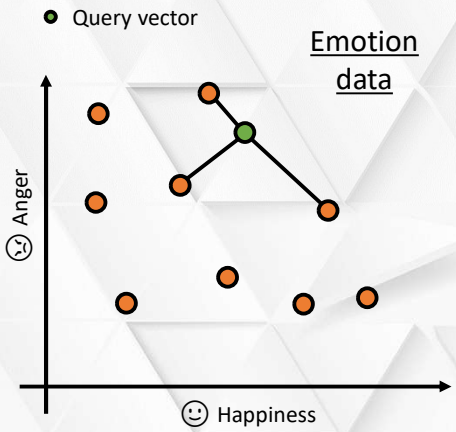
- Popular animation technique used for character locomotion
- Previously presented at GDC
  - **Motion Matching and The Road to Next-Gen Animation** *(Simon Clavet, GDC 16)*
  - **Motion Matching, The Future of Games Animation… Today** *(Kristjan Zadziuk, GDC 16)*
  - **Character Control with Neural Networks and Machine Learning** *(Daniel Holden, GDC 18)*
  - **Machine Learning Summit: Ragdoll Motion Matching** *(Simon Clavet, GDC 20)*
  - …

Characters locomotion was made with motion matching, a well-known technique previously presented at GDC. I do not intend to give a talk about motion matching, these guys do it much better than me. But briefly, the idea is that instead of building complex animation graphs, motion matching lets you find pieces of animation from a mocap dataset. Then, we can play animation to move the player character from its current pose to a desired one. For instance, here in Star-Lord's character, we can see in blue the desired "future" pose that is fed into the system and as a result, we see in green the best candidate that matches the input and in

purple, there’s another discarded candidate.

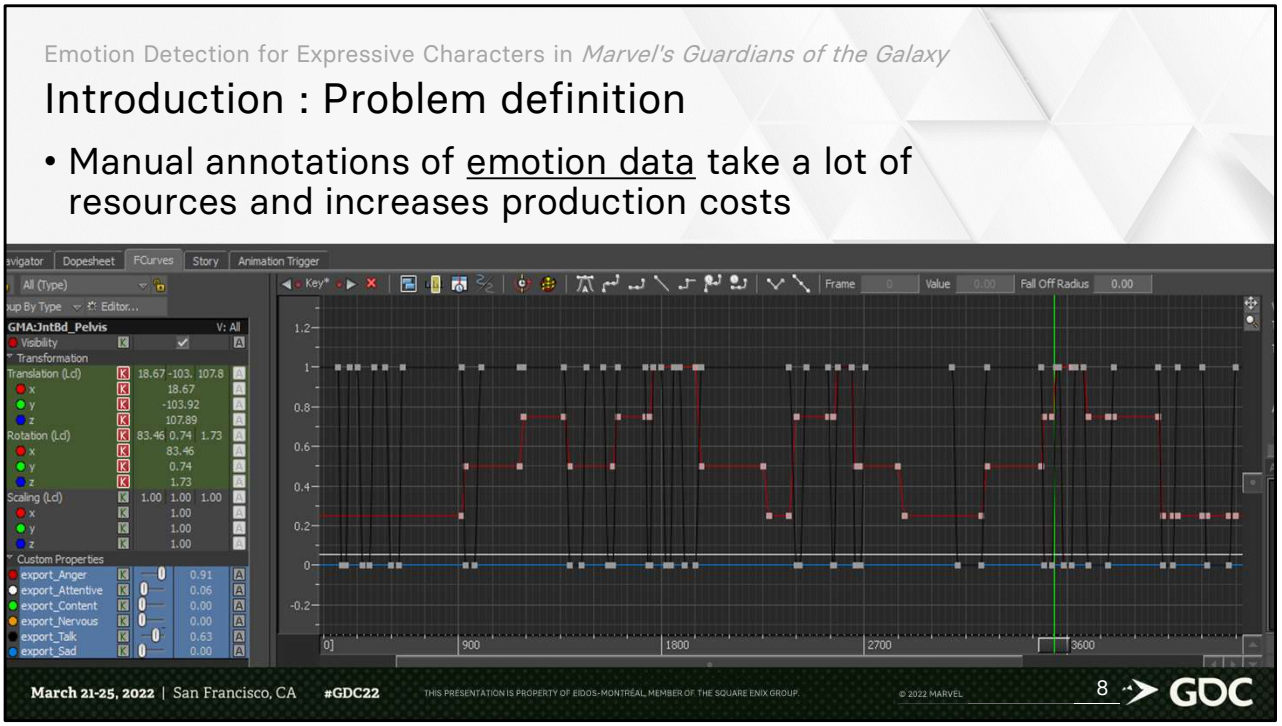
Introduction : From locomotion to emotion

- In house (e)motion matching system developed by production teams
- Query vector contains the body pose and the emotion data
- Responsiveness factor to mix body gesture with characters emotion
- Integrated to the animation sequencer for offline montage of animation sequences



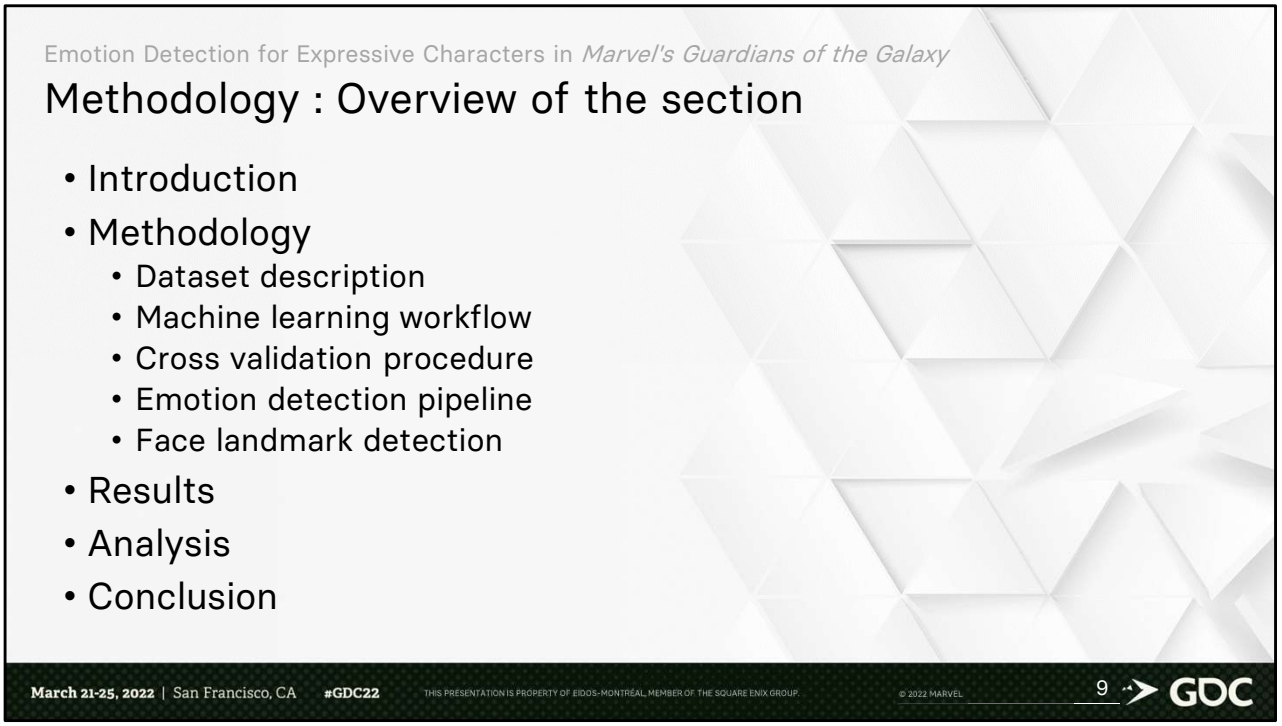
The production teams had their own motion matching system, and it was extended to work also with emotion data. When looking for an animation clip, the query vector contained the future pose and the desired emotion data. Such that they could look for any facial animation, with let's say, the actor showing an angry emotion and use this animation clip to make expressive dialogue between the characters. The facial animation and actor's voice were played back together, it was mixed with body gestures to better express character emotions. The emotion matching system was running

“offline” such that the animation staff could be playing and editing the animation content directly in the engine. Unlike character locomotion, the motion matching system doesn’t need to be responsive and match the player input.

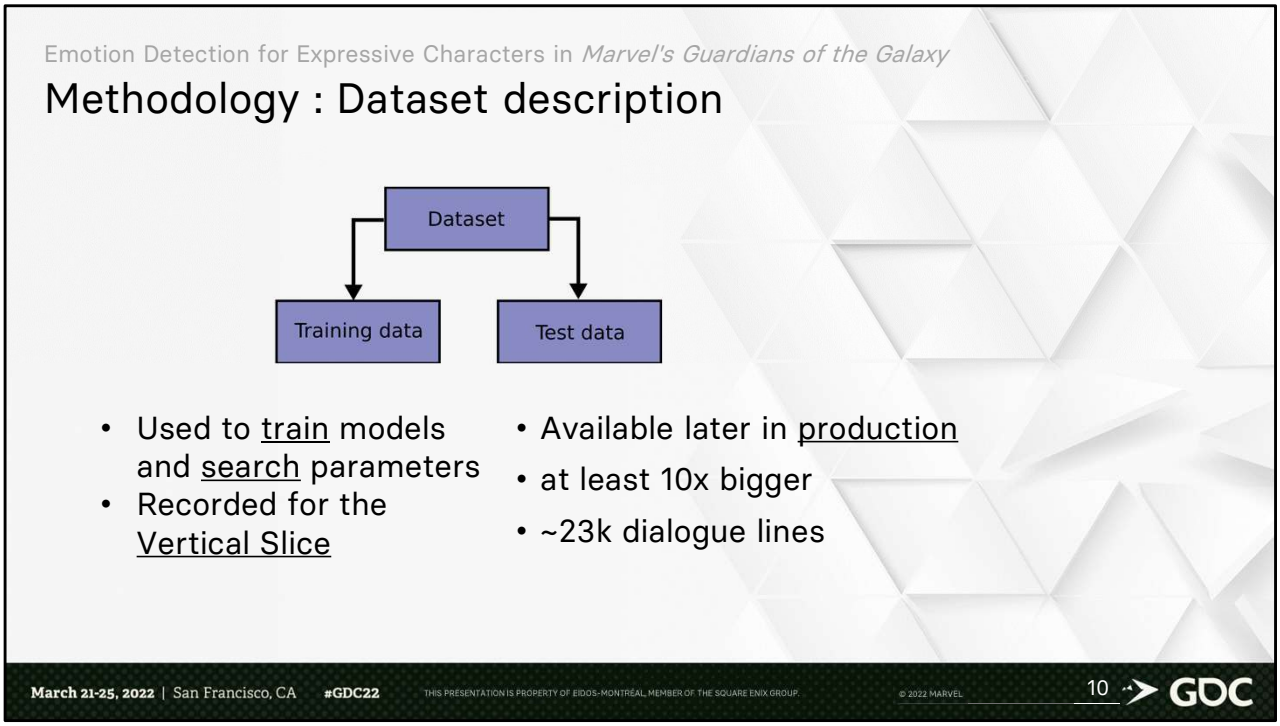


The dataset used by the emotion matching system was a set FCurve like this one, we can see the different tracks with actor's emotions in the bottom left corner of the screenshot, here I think it's in motion builder. It was made by a senior animator who watched a face capture video and set keyframe manually for several tracks, you can imagine how long it took to do it... It was a big problem for production because it costs a lot of resources and, well it could have cut the emotion matching technique from the game development. Hopefully, we can solve this problem with machine

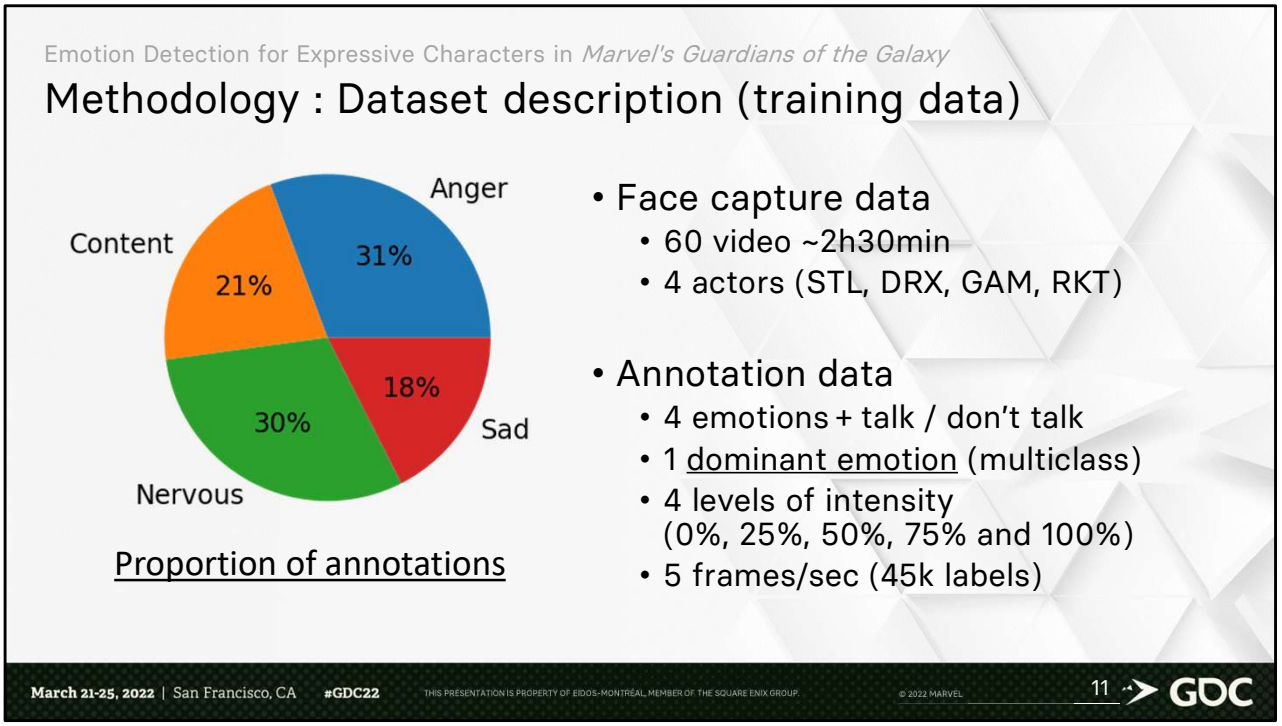
learning! These emotion curves can be baked automatically from the analysis of actor's facial expressions in face capture video.



In this section, I'm going to describe the dataset used to train machine learning models. I will also describe the workflow used to collaborate with production. I will finish by a brief description of the emotion detection pipeline used to bake emotion data.

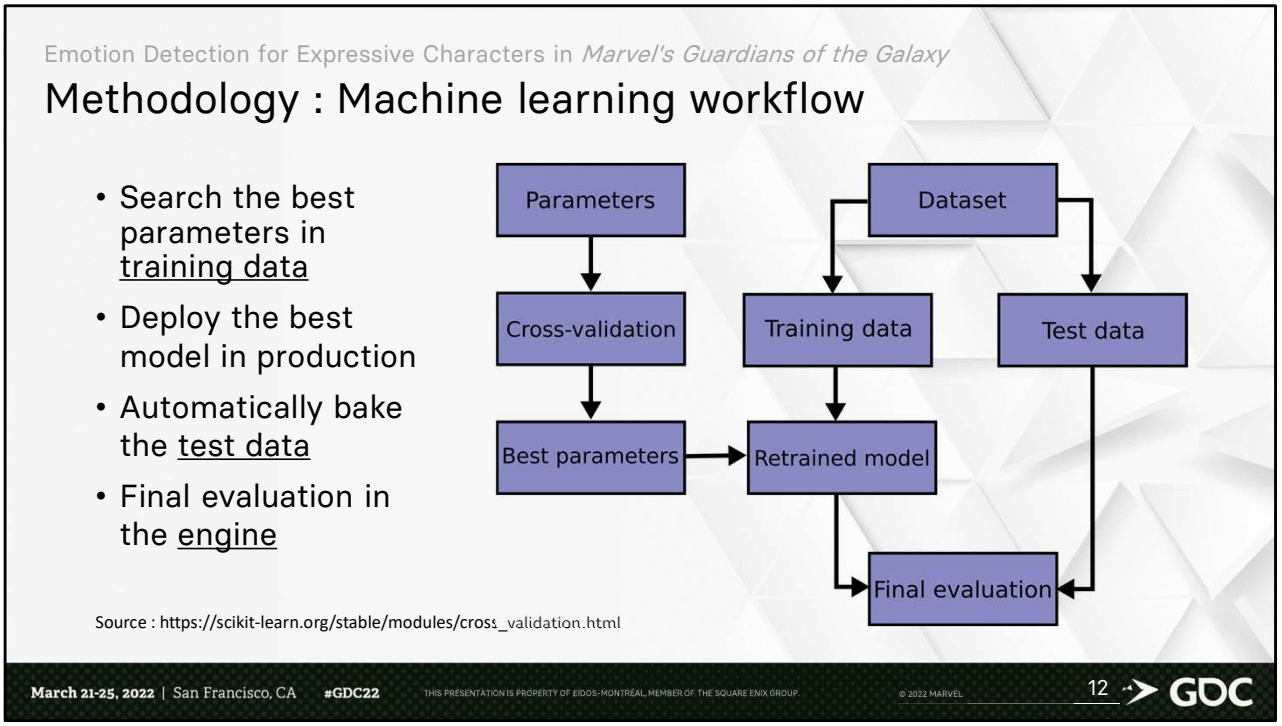


The dataset was split into a training set and a test set; this is a common operation in machine learning. The training set was used to train models and tune their parameters. It was recorded to make animation assets for the Vertical Slice of the game, while the test set wasn't available when I started the project. It was recorded later in production, during the game development process. Something important to have in mind is that it represents at least 10x times more data than the training set. To give an idea, there's something like 15 thousand dialogue lines in the game and most of them were not recorded yet when I did the project.

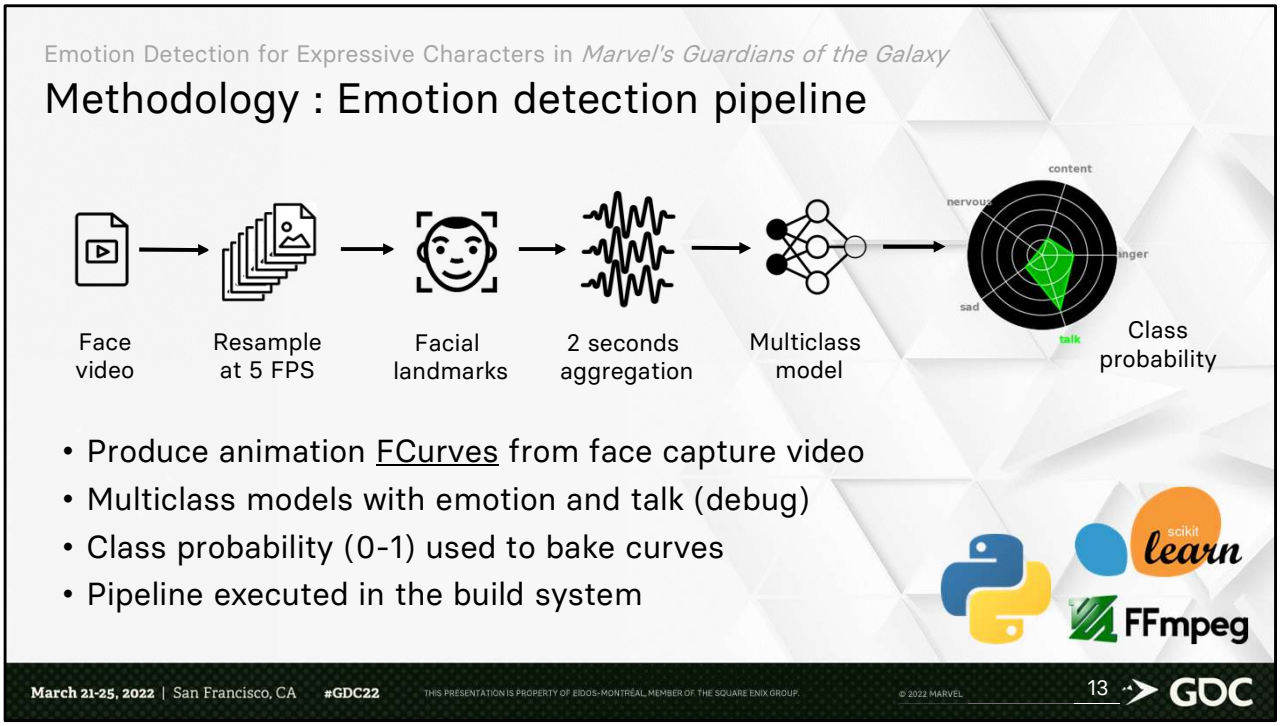


The training set was composed of face capture video, like the video I showed at the beginning of the presentation with the face of each actor recorded with a head mounted camera. In total, there were 60 videos for around 2h30min of data. Each video was annotated with 4 emotions (anger, content in the sense happy, nervous or sad). It was also annotated to know when the actor is talking or not. The animation team wanted to have 5 levels of intensity sampled at 5 frames per second, which makes a total of 45 thousand samples. That's the kind of curve I showed 3 slides ago in the problem definition

section. These annotations were available in the training data but not in the test data. With one dominant emotion at a time (the actor cannot be happy and sad at the same time), it gave me an opportunity to train multiclass models with supervised learning.

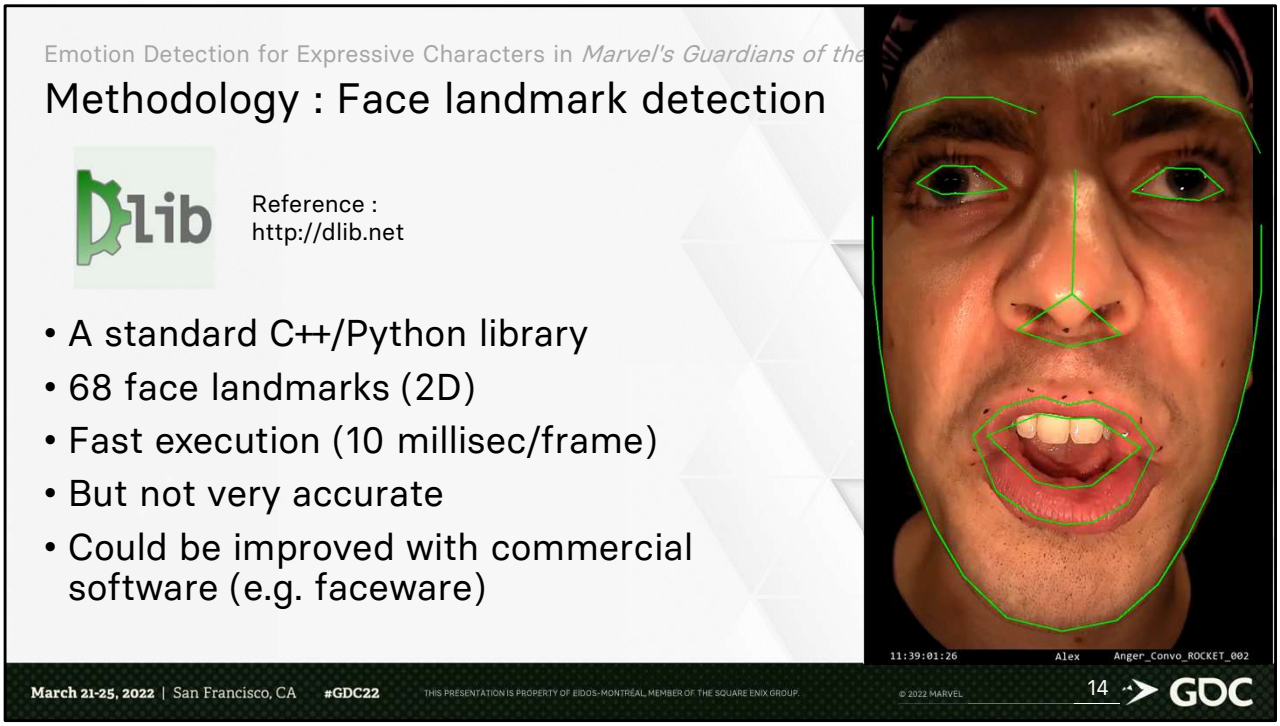


This is our machine learning workflow. You probably know that it can be tricky to train machine learning models because of the noise and biases in datasets, notably with human annotations. So, I used parameter search techniques with cross-validation to find the best parameters in the training data. The best model was deployed in the build system for final evaluation by the animation team in the game engine. It was used to bake the animation curves automatically, after each build, including the latest data recorded for the game development.

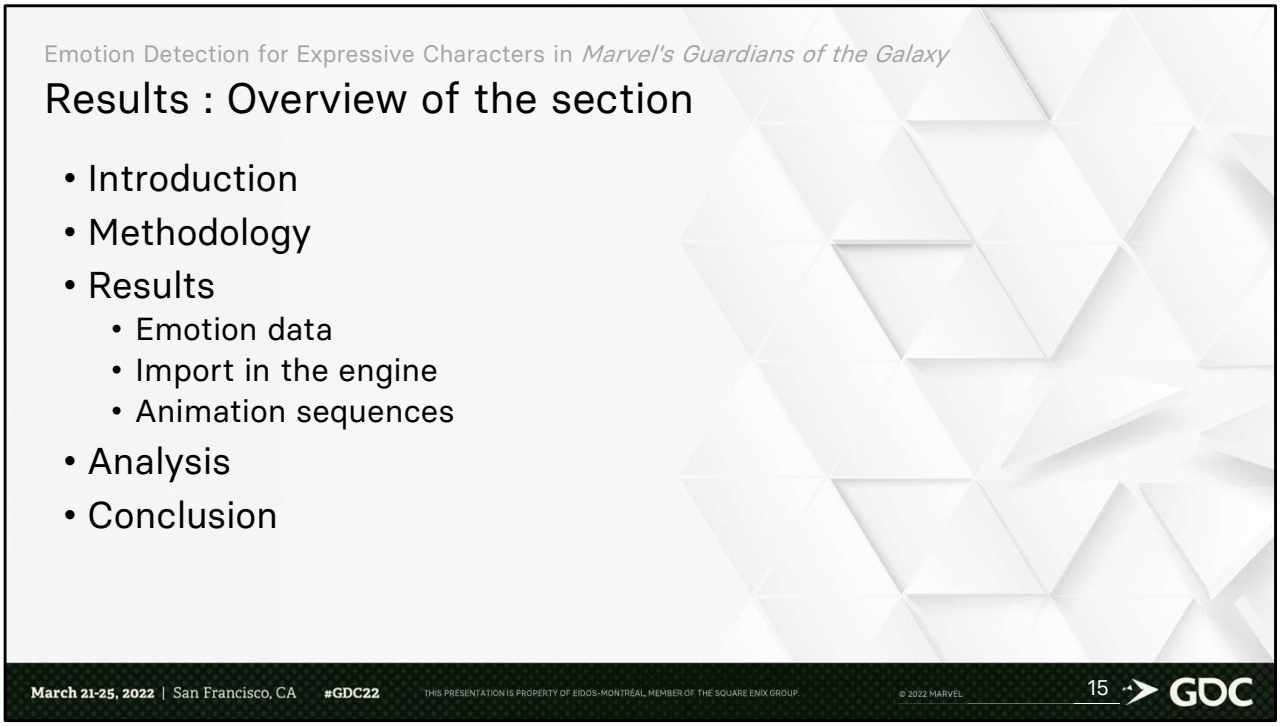


The goal of the emotion detection pipeline was to compute animation curves directly from the actor's facial expressions. It was implemented in python for compatibility between tools used by machine learning and game production. Each face capture video was resampled at 5 frames per second to match the sampling frequency requested by the animation staff. It seems fast enough to capture most of facial expressions, maybe not micro-expressions… Then facial landmarks were extracted from the actor's faces, I'll show an example in the following slide, that's the kind of markers used in face rigs for retargeting the actor's expressions to the

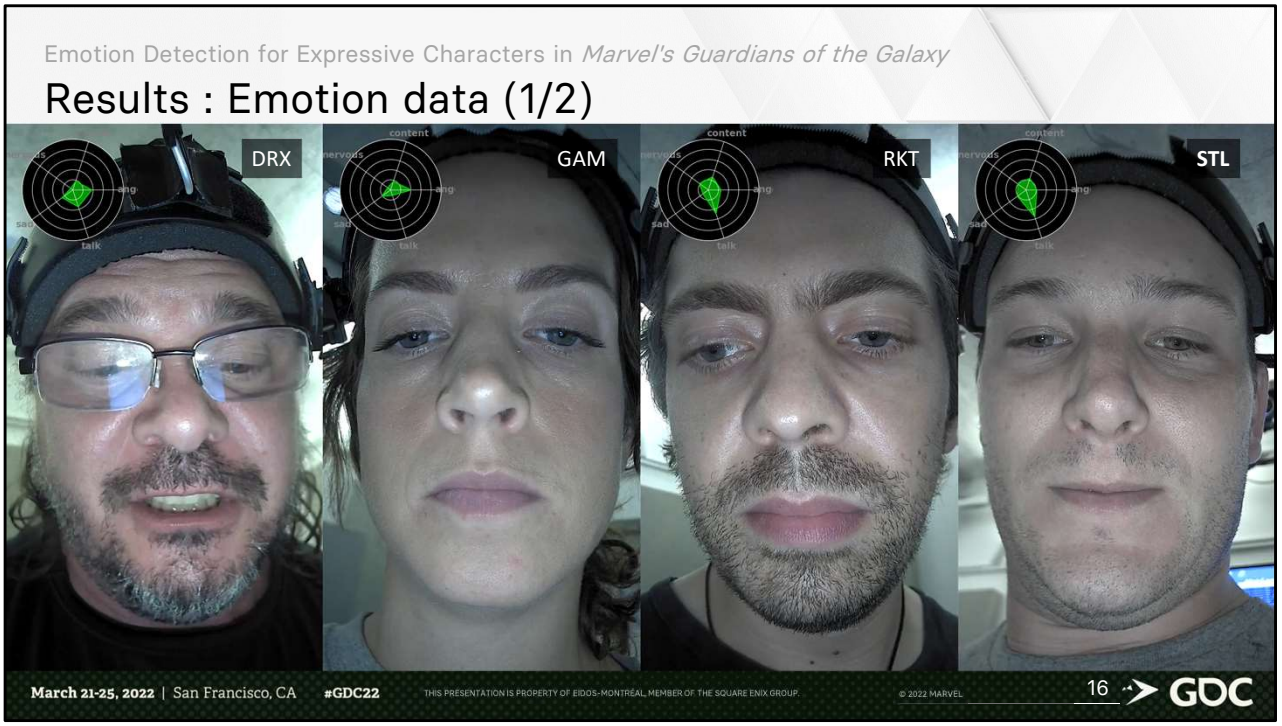
virtual character. I thought it would be more robust to detect emotions from these kinds of features instead of video images. It was also easier to aggregate these features on a given time window, here I choose 2 seconds to feed the model with enough data, which really helps with such complex face movements. Finally, these features are normalized and fed as input to multi-class models to detect emotions. I also included the class talk / don't talk for debug purposes; it turns out that production used audio speech for making this kind of animation track. The model outputs a probability between 0 and 1 which is used to make animation curves for each emotion. I tried different types of models from the scikit-learn toolbox which provides a very flexible environment for machine learning research.



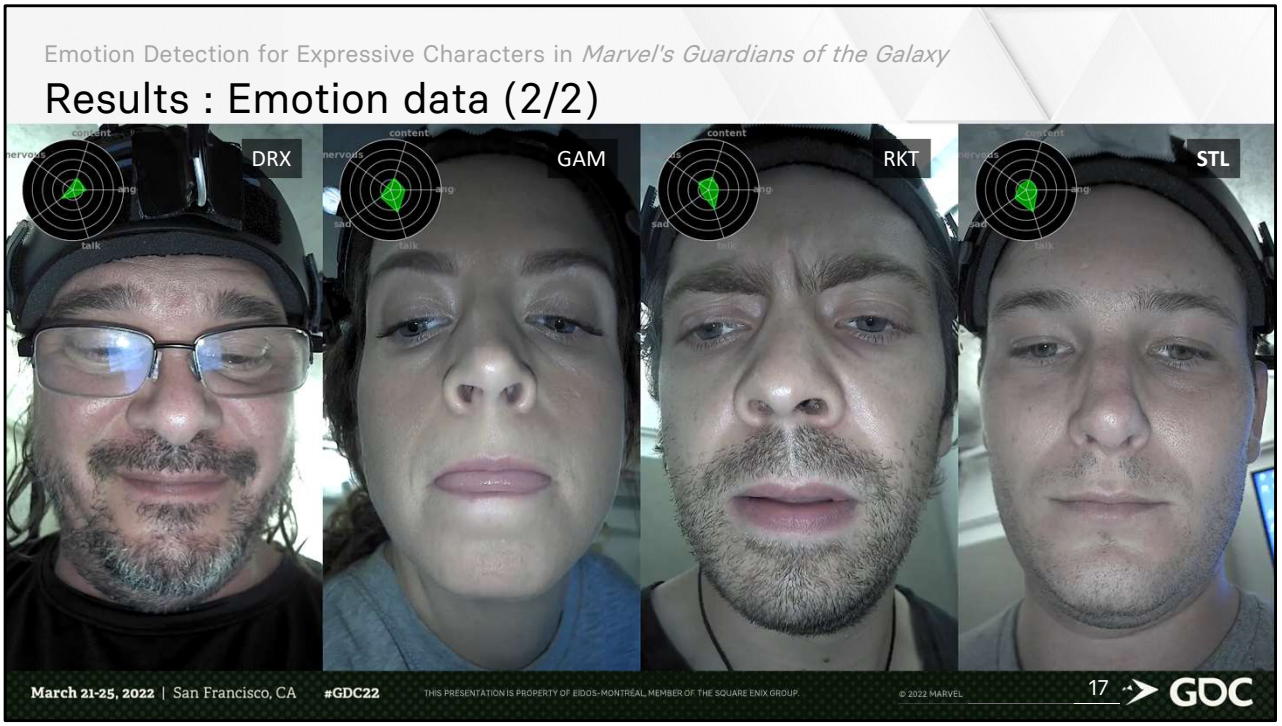
I'd like to spend one moment on the face landmarks detection, because when I did the project, I didn't know anything about facial animation, so I used dlib, one of the most standard libraries in C++/Python. For each frame of a face capture video, like this one, it computes 68 2D face landmarks overlayed in green here. It's fast to execute on a standard workstation but it's not very accurate, as we can see sometimes, the landmarks are not very well aligned with facial movements. This point could be improved with a software like Faceware which production did actually use, but I wasn't aware of it during the project.



Now it’s time to show some results. I’ll start with an example of emotion data extracted from the test data with the emotion detection pipeline. Then, I’ll show how these data were imported in the game engine and I’ll show an animation sequence obtained with the emotion matching system.

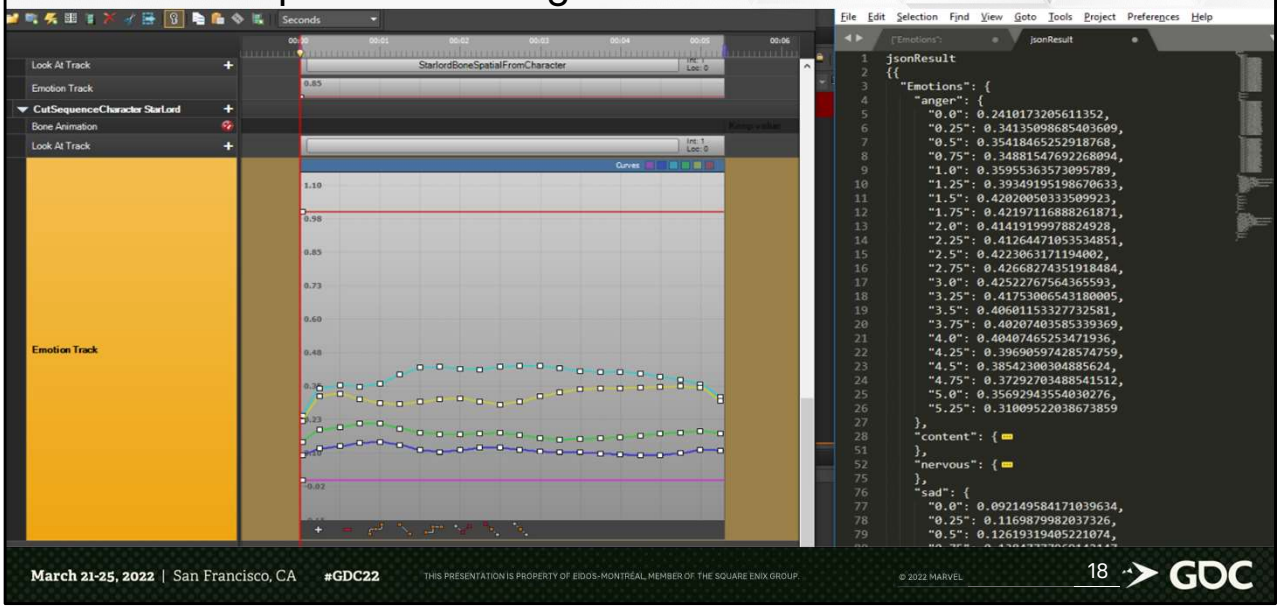


So, this is a set of face capture videos recorded simultaneously. You'll see the emotion data overlayed in the small radar plot, in the top left corner of each video



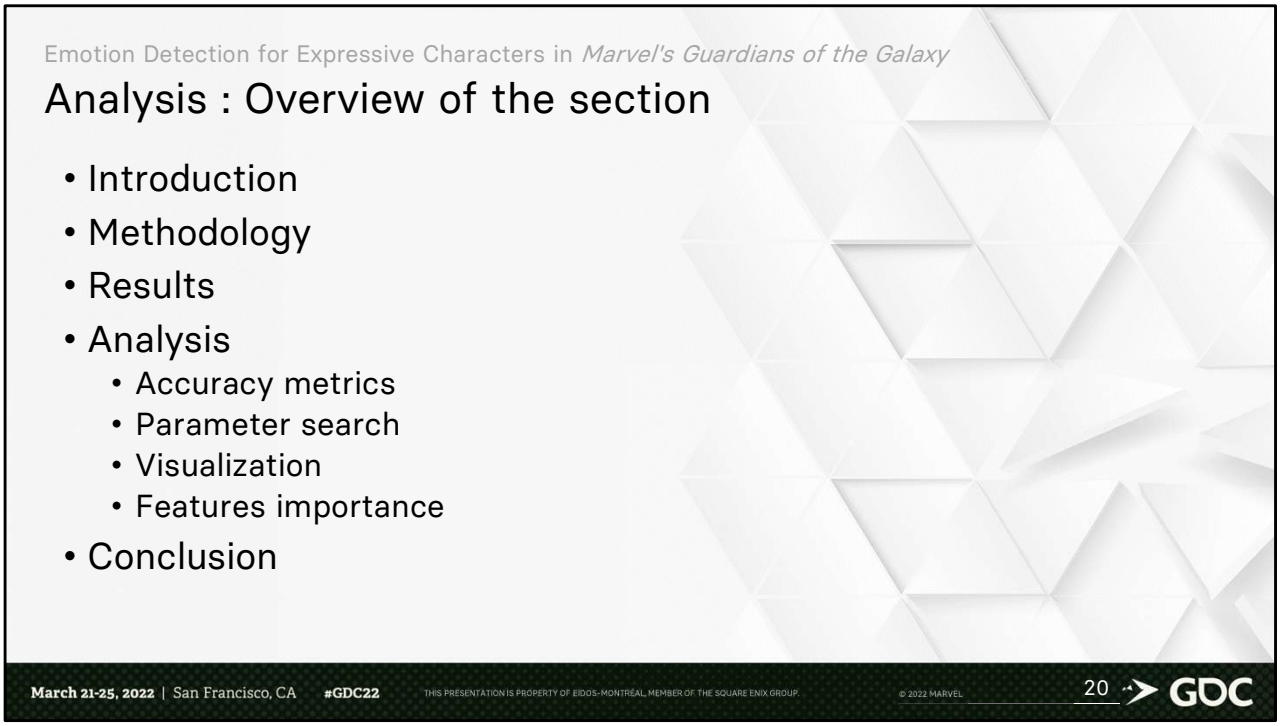
here is another example video

Results : Import in the engine

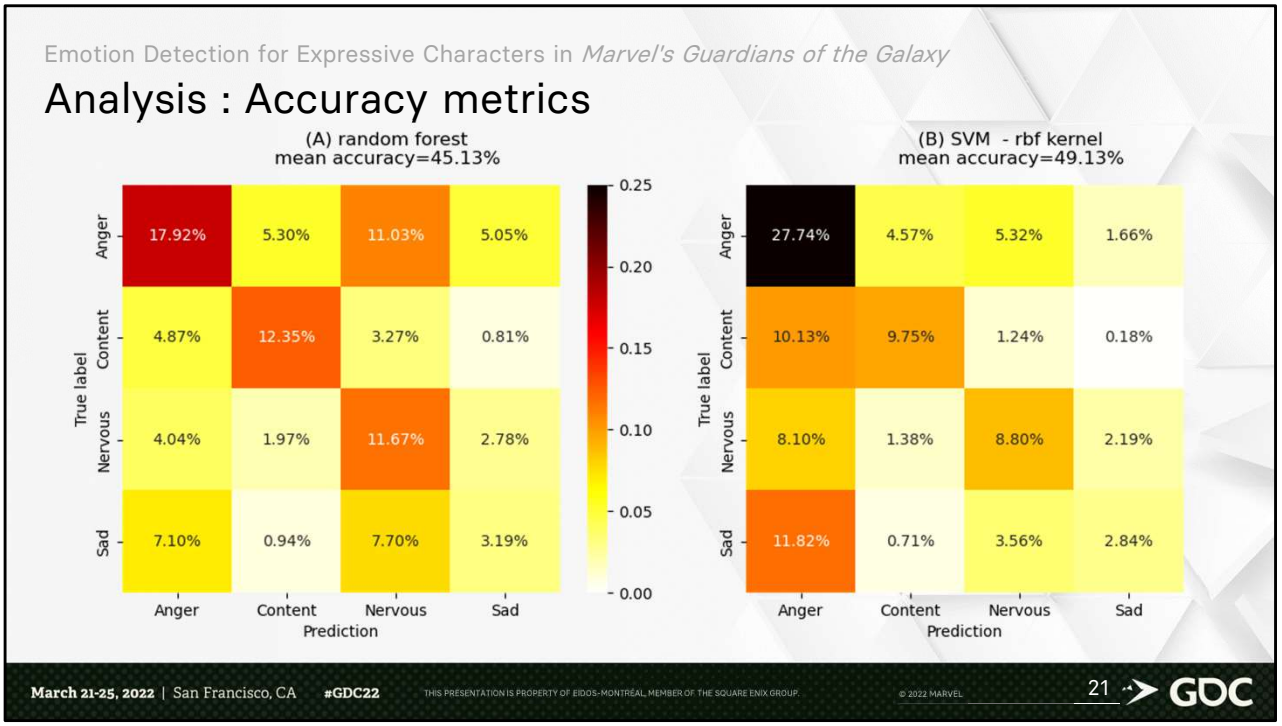


I know people at GDC love to see results in the game engine. Here is a screen shot of the emotion data imported from model output to the emotion track of the animation sequencer in the studio game engine. On the right side of the screen, we can see these data into a json file. There's a set of keyframe and model output formatted as a dictionary, it was generated by the emotion detection pipeline.

Here it’s a sequence of animation obtained with emotion matching [START VIDEO] because it is scripted offline, it can be played in another situations like in traversal gameplay.

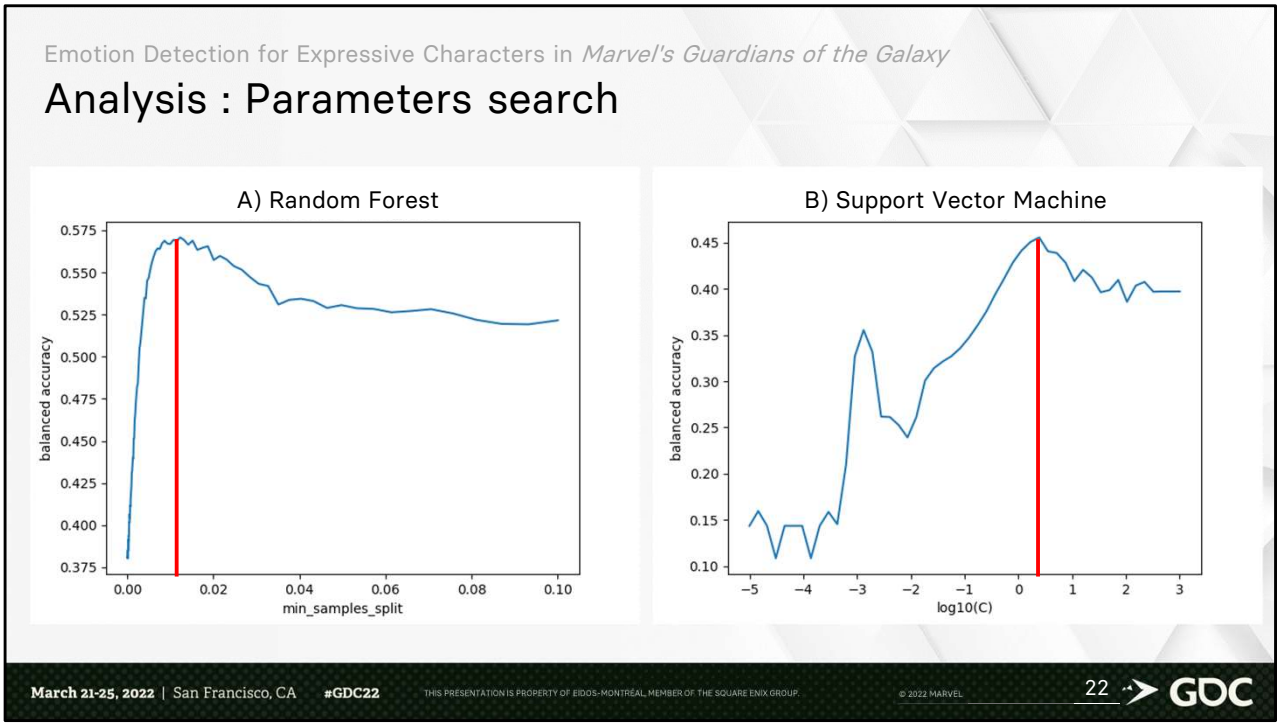


In this analysis section, we'll look at accuracy metrics obtained with cross-validation, and we'll see some parameter search results. Finally, I would like to show some tools for interpretation of the results, it is often said that machine learning models are black boxes... we'll see that it depends on the model and on the dataset.

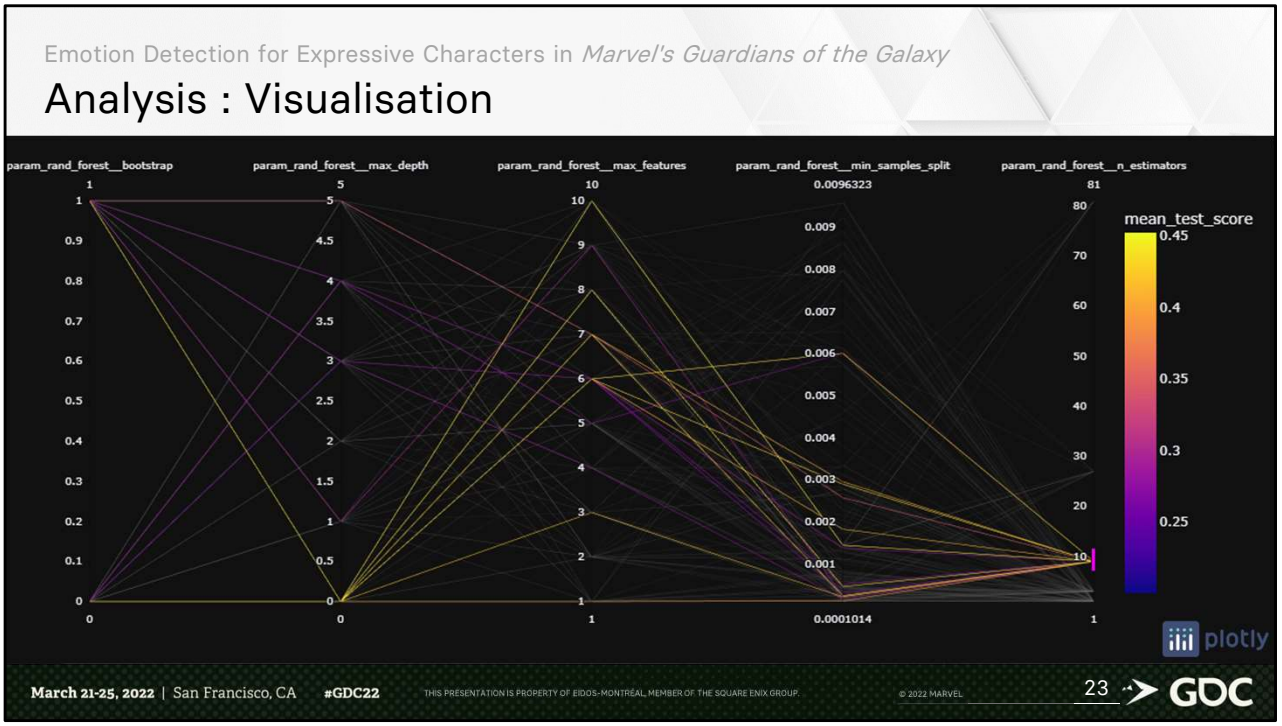


If we'd like to have a closer look at the accuracy metrics, here, we have the confusion matrix for a Random Forest (A) on the left and for a Support Vector Machine (B) on the right. In the vertical axis we have the true label, the annotation provided by the animation staff and in the horizontal axis, we have the model predictions. The elements on the diagonal are the proportion of frame with correct classification. Interestingly, the random forest model often makes the confusion between anger and nervous emotions. The SVM model on the right, however, even if its average classification accuracy is

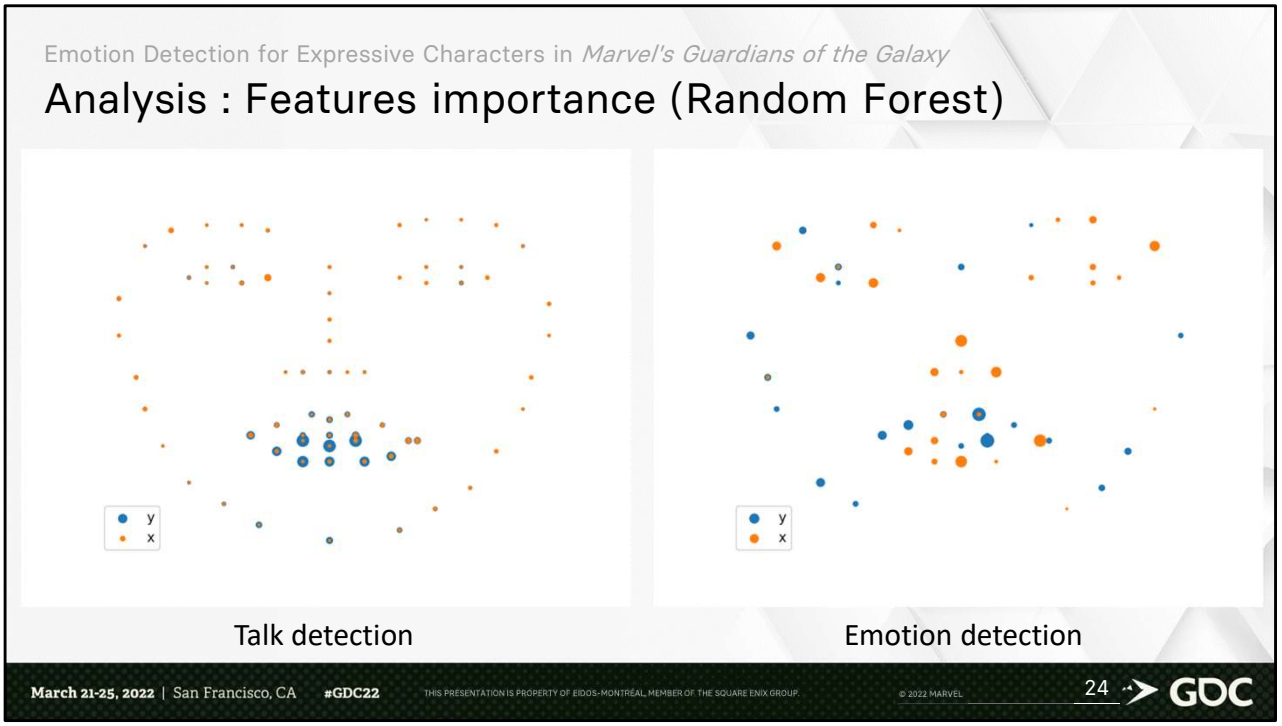
higher than the random forest model, it sounds less reliable because its predictions are biased towards anger. This is probably due to overfitting the training data, the model is catching that anger is frequently annotated and make more predictions about it.



In order to avoid these kinds of issues, we can use a metric called “balanced accuracy”. Here is a couple of tuning curves computed over a regularization parameter (min sample split, for random forest, and the parameter C for the SVM). The balanced accuracy is computed by cross-validation of the training data. This kind of technique allows you to find the best parameter of a model, indicated by the red bar. It can be extended to multiple parameters with grid search, but it can be quite expensive to compute when the number of parameters is large.

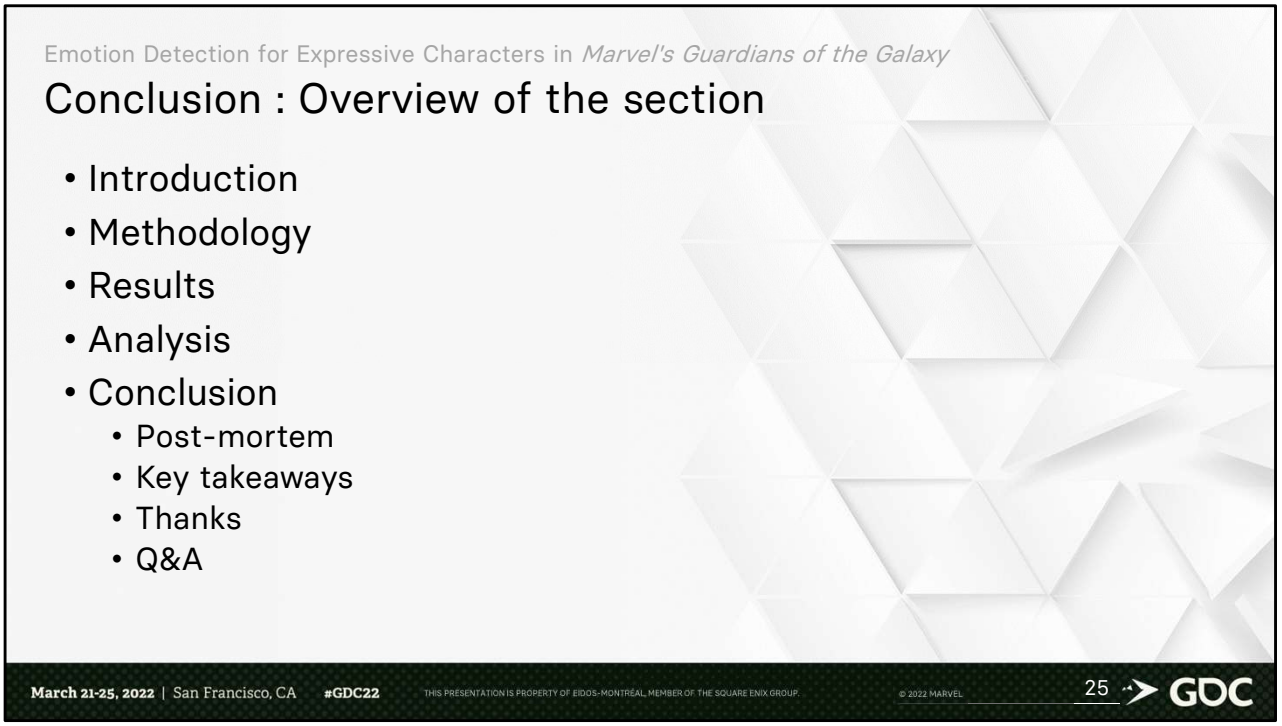


Another parameter search method used for this project was hyperband. It's a bandit-based algorithm with an interesting adaptive resource allocation mechanism. I used it to find the best parameters of a random forest classifier, while reducing the disk usage with n_estimator as resource parameter. Here we can see the result of the algorithm with a parallel coordinate plot using plotly. It shows the accuracy of the classifier in color with corresponding parameters that can be traced back, the best models often have a few estimators. This kind of visualization is useful when the searching over more than 3 or 4 parameters...

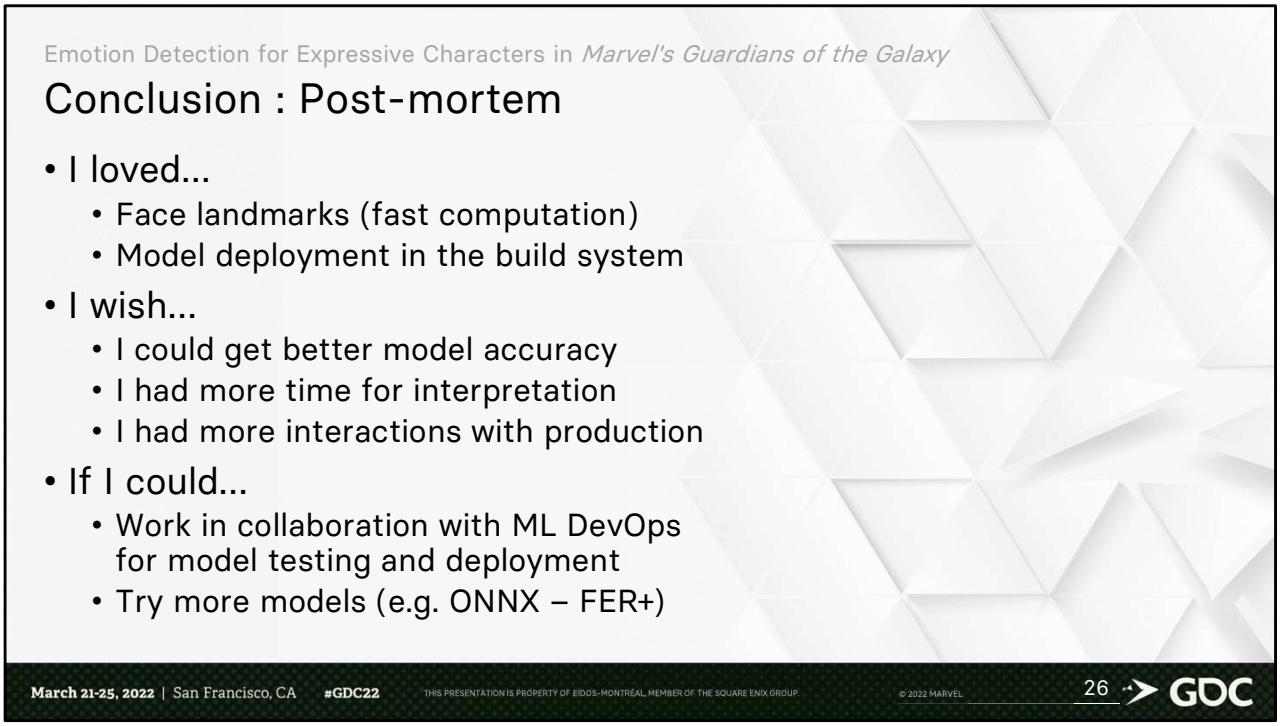


An interesting property of random forest models is their interpretation capability. It's often argued that machine learning models are considered as black boxes, well it depends. Here we can see the feature importance of a model trained for talk detection on the left and for emotion detection on the right. For talk detection task, the most important features are vertical locations of facial landmarks around the mouth. The model learned that the mouth is moving when the actor's talk, it is not surprising but that's a good sanity check... however, if we look at features importance for an emotion detection model, it doesn't provide a clear interpretation of the

results. That’s because patterns learned by the model in the emotion detection task are probably more complex than those in a simple task like talk detection.

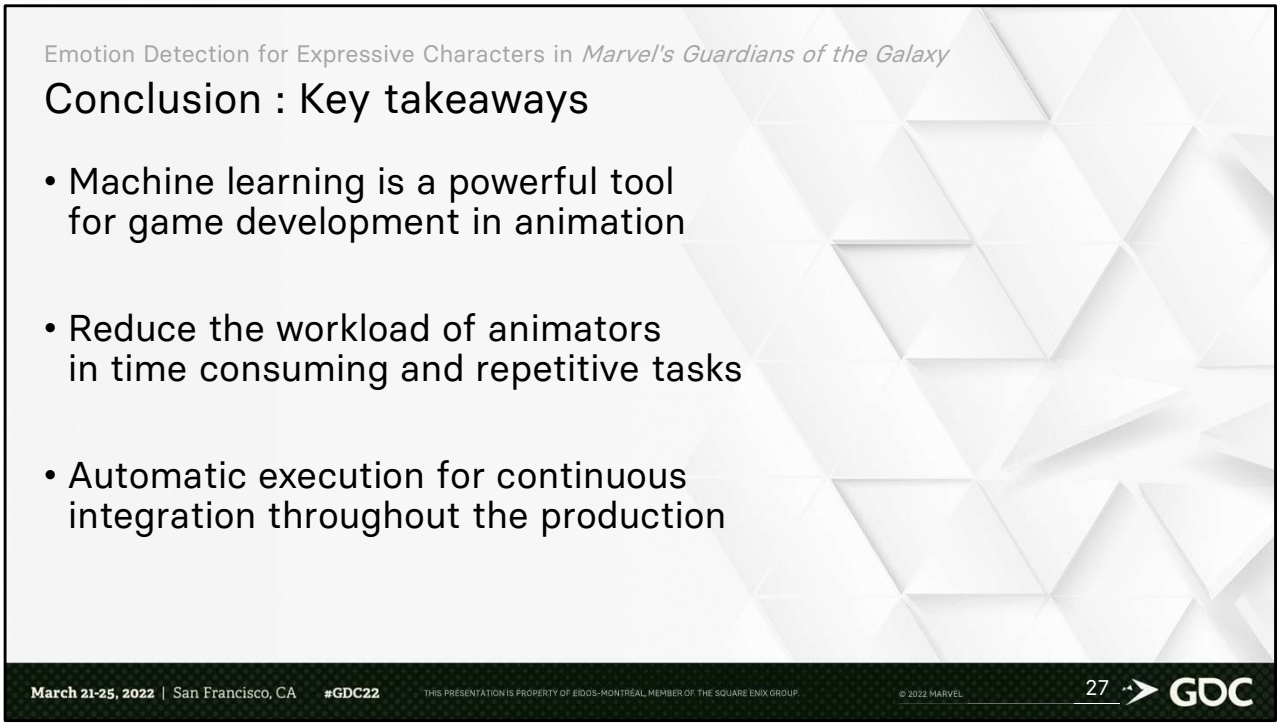


So, this last section is the conclusion of my presentation.



As a post-mortem examination of my project, I would like to say that it's important to find a good trade-off between model interpretation and performance. Searching for the best parameters is useful but the results should always be interpreted carefully. I wish we could have more time for tweaking parameters and get more feedback from production, but we didn't. I think we deployed 2 or 3 different models in the build system. Once the game production started and the animation staff was happy with the model, we didn't retrain any model because people were busy somewhere else. I also wish I could work in closer collaboration with DevOps for

automatic model training and deployment, but we didn’t need to... I was just uploading my work to a GitLab repository that was pulled from the build system and voila, we didn’t build a docker image or provide any API services...



Here are some takeaways: Machine Learning is a powerful tool for game development. In animation, it can be used to drive animation sequences with emotion data, directly extracted from face capture video. In Marvel's Guardians of the Galaxy, we leveraged machine learning to ship the game with a new animation technique called "emotion matching". This technique became production ready because machine learning reduced the time spent on time consuming and repetitive tasks, such as the annotation of face capture video. Even if the accuracy wasn't perfect, the animation team reported that a very large portion of emotion

curves post-processing was done by the emotion detection pipeline, only a few edge cases needed some manual intervention. Another important takeaway is that the pipeline was deployed in the build system of the game, it was baking emotion curves from face capture video. Machine learning was not executed at runtime because the animation staff wanted to script the animation sequence offline, it was also safer and less computationally expensive.

Conclusion : Thanks

1. Dominic Duchemin
   Principal Animation Programmer
2. Sébastien Proulx
   Technical Director Animation
3. Julien Groulx
   Animation Programmer
4. Francine Mangin
   - Senior Animator
5. Pierre-Claude Jacob
   Senior Tools Programmer
6. Eric Martel
   - AI / ML Director

To finish, I would like to thank people who worked with me on this project. Dominic Duchemin who developed the emotion matching technique, Seb Proulx who was leading the project, Julien Groulx who was doing the final evaluation of my work in the engine, he also helped me a lot with this presentation. I would like to thank Francine who did the annotation work and Pierre-Claude who did the tooling part for the integration of my work. And finally, Eric Martel who gave me the opportunity to work on this project, he was the AI/ML director at Eidos Montreal and my boss.

Emotion Detection for Expressive Characters in *Marvel's Guardians of the Galaxy*

Conclusion : Q&A

trachelr

Romain Trachel

@RomainTrach

Thanks for watching my presentation. Feel free to contact me for questions. Bye-bye!