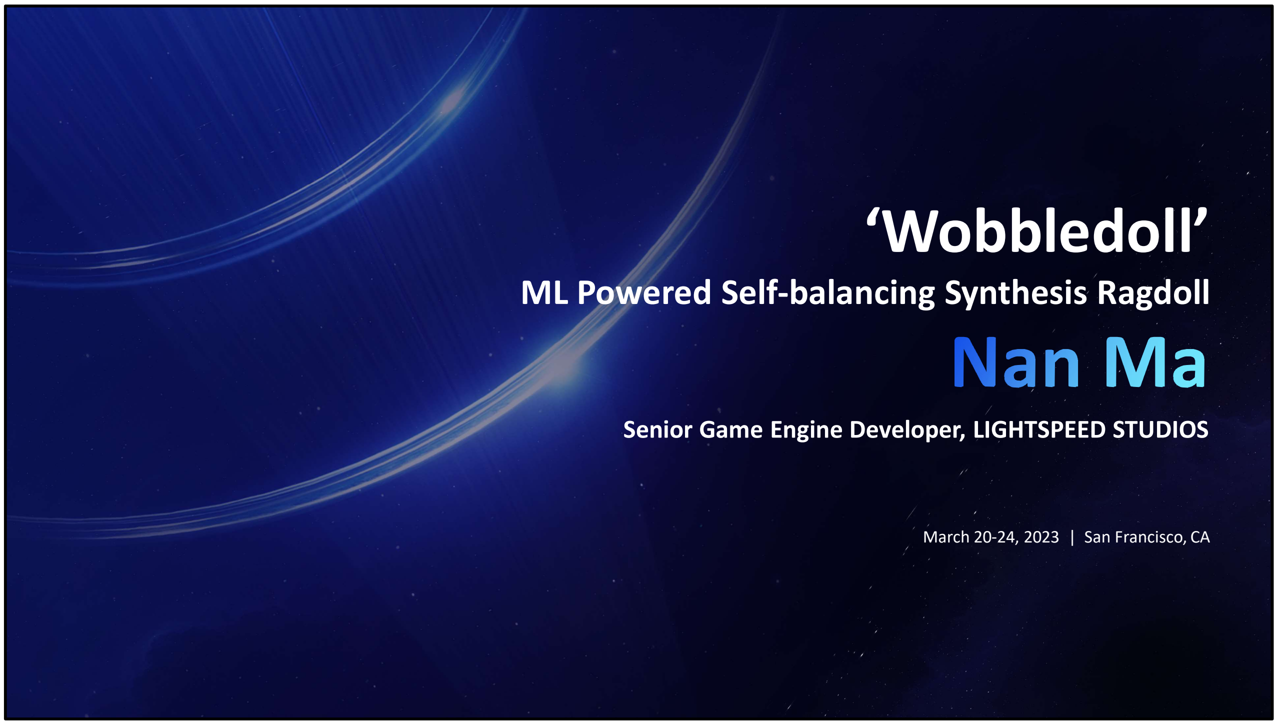
MEET LIGHTSPEED STUDIOS AT GDC2023

March 20-24, 2023 | San Francisco, CA



‘Wobbledoll’

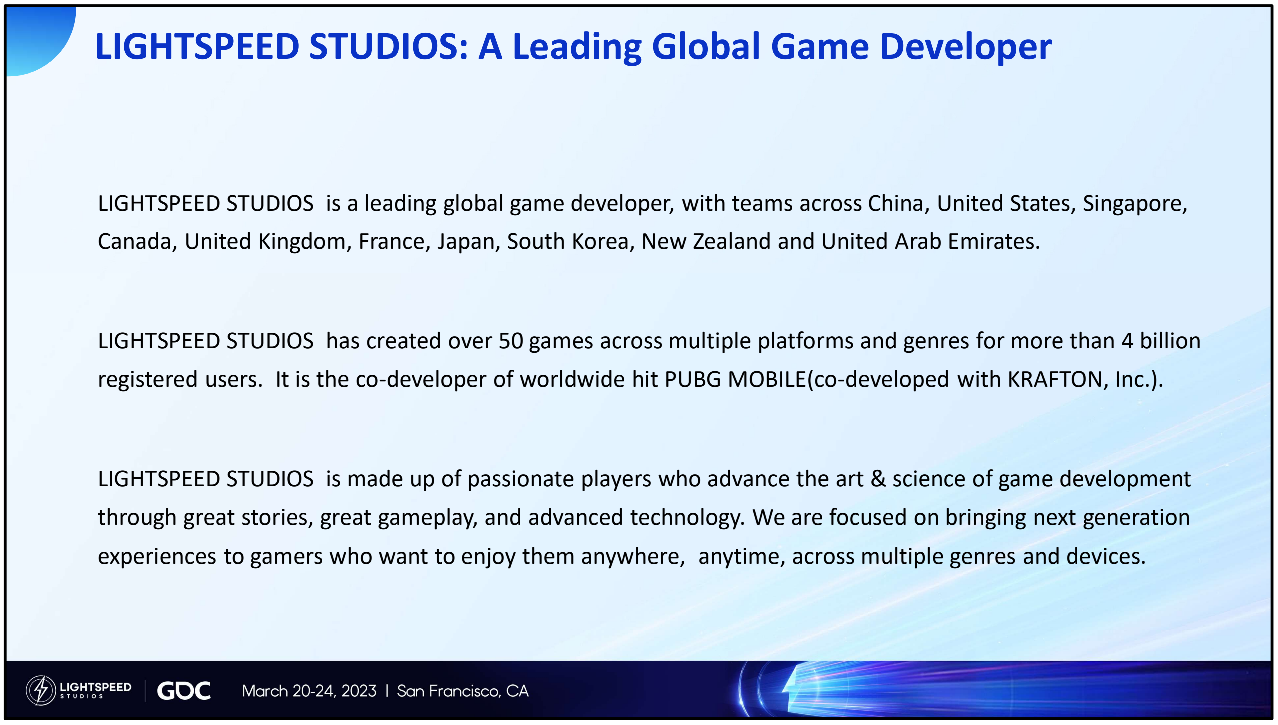
ML Powered Self-balancing Synthesis Ragdoll

Nan Ma

Senior Game Engine Developer, LIGHTSPEED STUDIOS

March 20-24, 2023 | San Francisco, CA

Hello everyone, this is Nan Ma, I’m senior game engine developer from Lightspeed Studios. It is great pleasure to present the Wobbledoll system to you, which is the smart self-balancing ragdoll that’s powered by the machine learning.



LIGHTSPEED STUDIOS: A Leading Global Game Developer

LIGHTSPEED STUDIOS is a leading global game developer, with teams across China, United States, Singapore, Canada, United Kingdom, France, Japan, South Korea, New Zealand and United Arab Emirates.

LIGHTSPEED STUDIOS has created over 50 games across multiple platforms and genres for more than 4 billion registered users. It is the co-developer of worldwide hit PUBG MOBILE(co-developed with KRAFTON, Inc.).

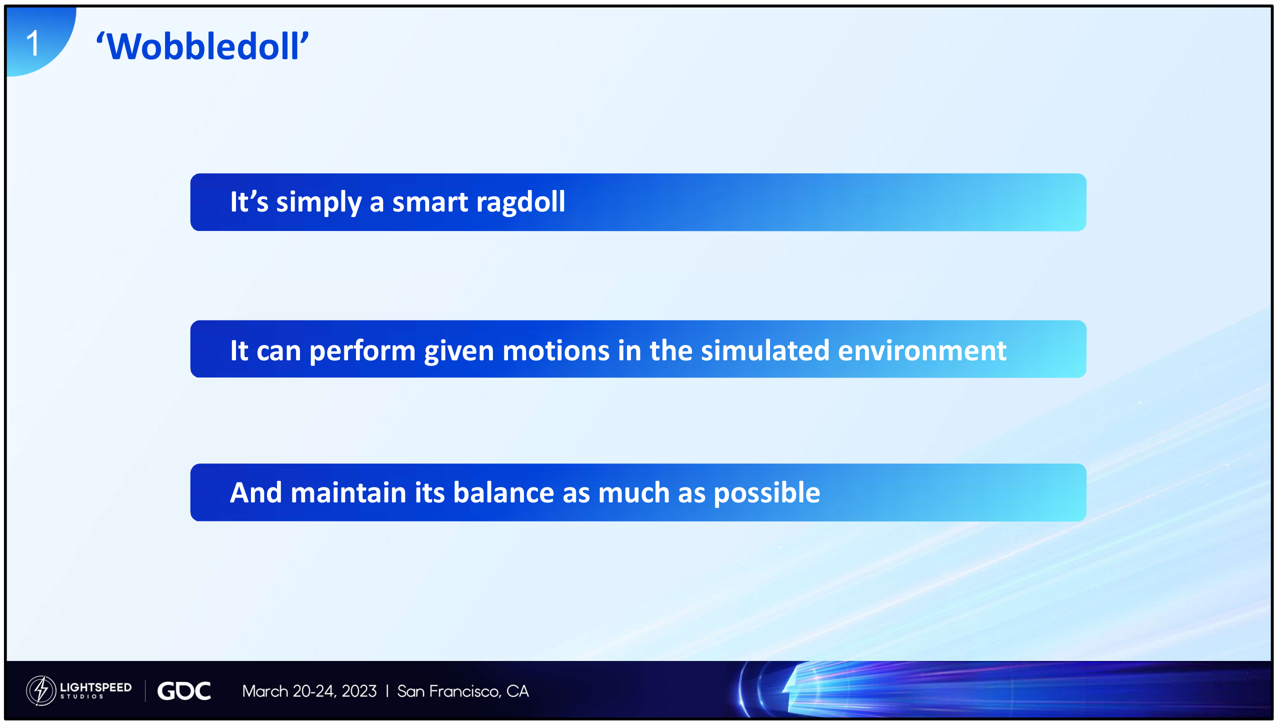
LIGHTSPEED STUDIOS is made up of passionate players who advance the art & science of game development through great stories, great gameplay, and advanced technology. We are focused on bringing next generation experiences to gamers who want to enjoy them anywhere, anytime, across multiple genres and devices.

LIGHTSPEED STUDIOS | GDC March 20-24, 2023 | San Francisco, CA

Just a bit background information about Lightspeed Studios for people haven't heard about us.

Lightspeed Studios is a leading global game developer, with teams across all over the world. And myself from the Canada team.

Our studio has created over 50 games across multiple platforms and genres. One of the game you guys probably know is the PUBG Mobile


1 ‘Wobbledoll’

It’s simply a smart ragdoll

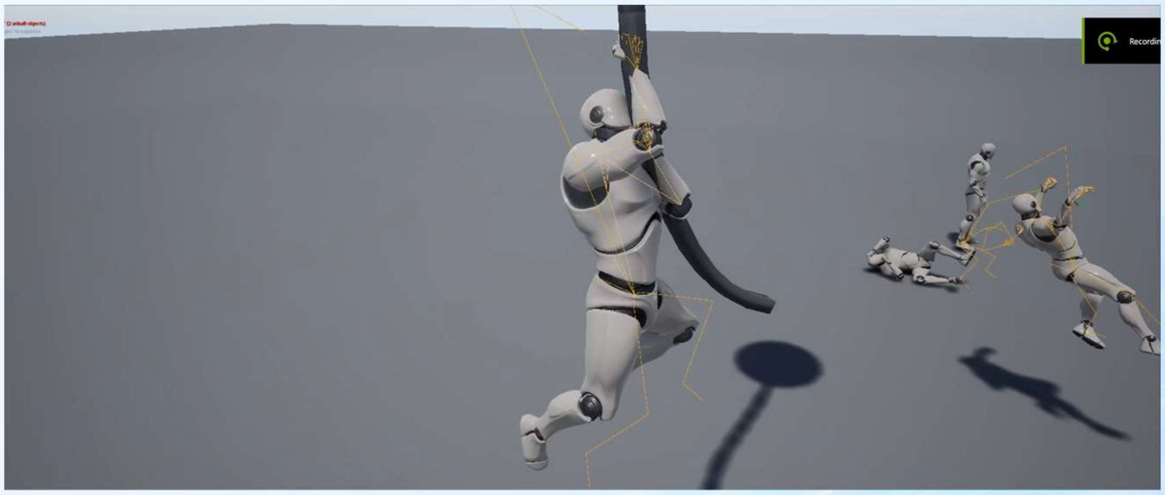
It can perform given motions in the simulated environment

And maintain its balance as much as possible

LIGHTSPEED STUDIOS | GDC March 20-24, 2023 I San Francisco, CA

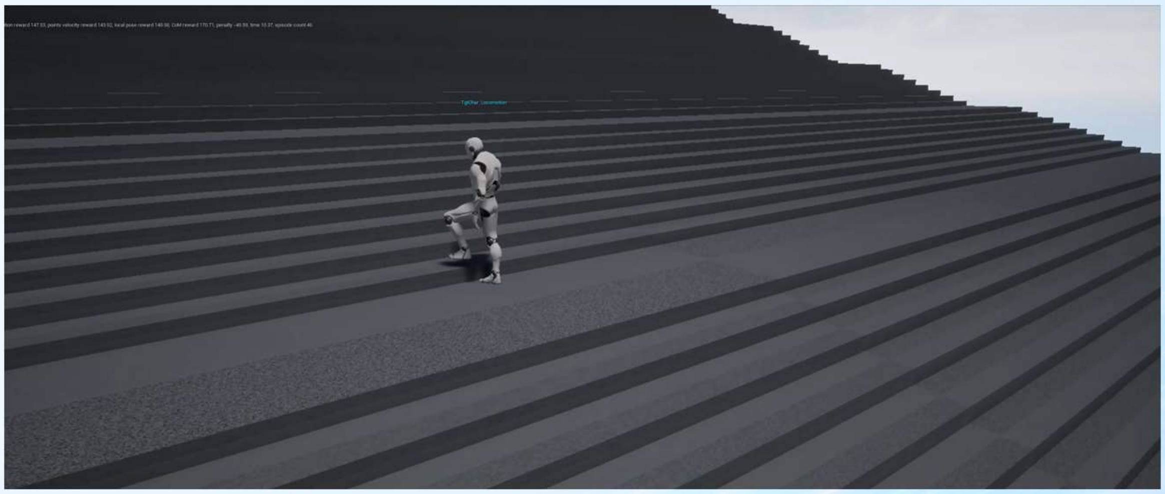
Alright back to our topic, what is the Wobbledoll. It’s basically an advance ragdoll that able to perform given actions while maintain the balance.
To get better understanding, let’s take a look at a few Wobbledoll demo videos

2 **Wobbledoll Performs Given Action**



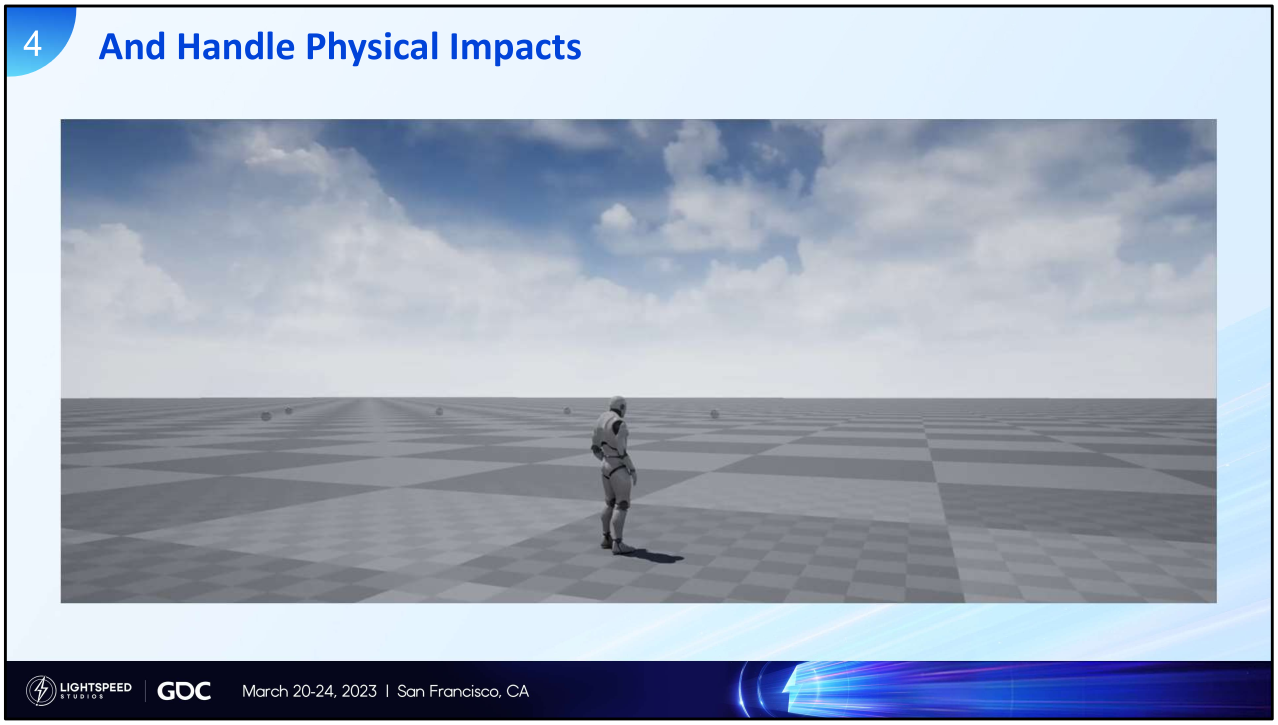
This video shows a Wobbledoll character climbing along the ropes. The yellow debug skeleton reveals the underlying animation the character is following. As you can see, the Wobbledoll character is able to perform a realistic rope climbing motion, interacting with the ball impacts and swinging with the rope. The motion looks natural and grounded, just as it should be in real life.

3 Maintain Self-balance



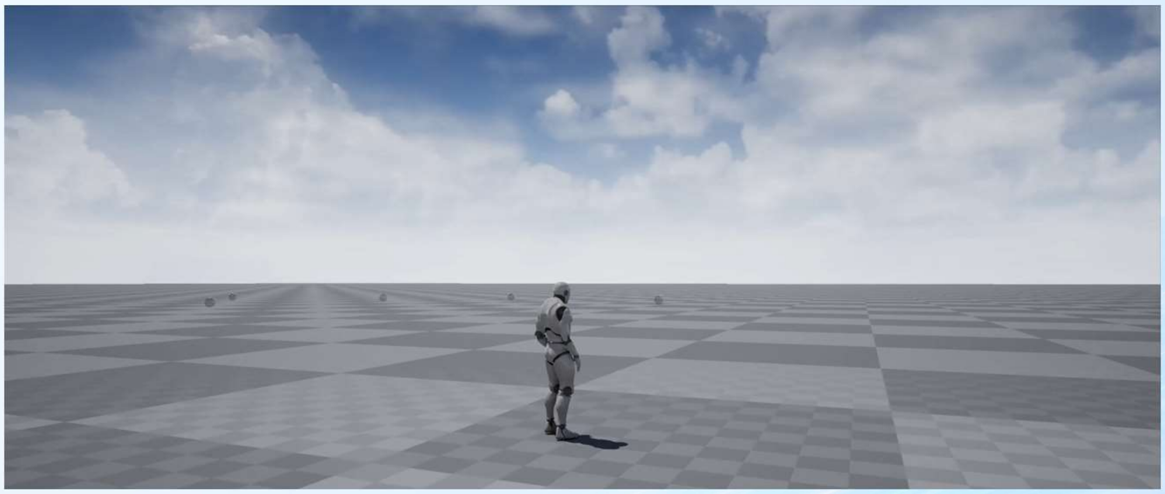
LIGHTSPEED STUDIOS | GDC March 20-24, 2023 | San Francisco, CA

Here in this demo, we have a Wobbledoll character walking up and down stairs. The Wobbledoll balancer is capable of handling complicated terrain such as stairs and slopes. You can see that the character’s foot slips down a stair occasionally, but it is still able to maintain its balance in those cases.



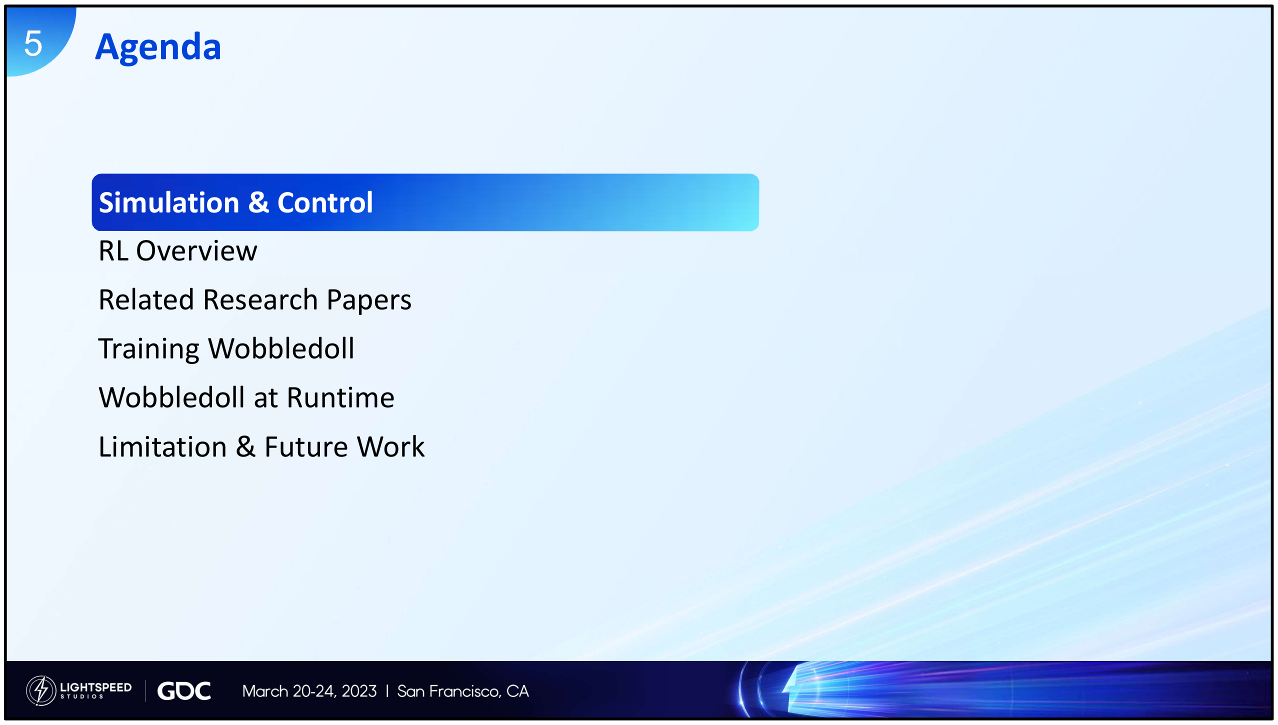
4

And Handle Physical Impacts



LIGHTSPEED STUDIOS | GDC March 20-24, 2023 | San Francisco, CA

This demo video here showcases how Wobbledoll system able to handle physical impact and still remain balanced. Each ball weights 10kg and launches at 20m/s. The Wobbledoll character is able to survive from such large impacts while standing still or in motion.



5

Agenda

Simulation & Control

RL Overview

Related Research Papers

Training Wobbledoll

Wobbledoll at Runtime

Limitation & Future Work

LIGHTSPEED STUDIOS | GDC March 20-24, 2023 | San Francisco, CA

In today's talk, I would like to take this opportunity to walk you through the journey of how we are utilizing ML to build our physical interaction system, called Wobbledoll.

Here’s an agenda for all the topics I’d like to cover.

## First thing I’d like to talk about is the simulation and control

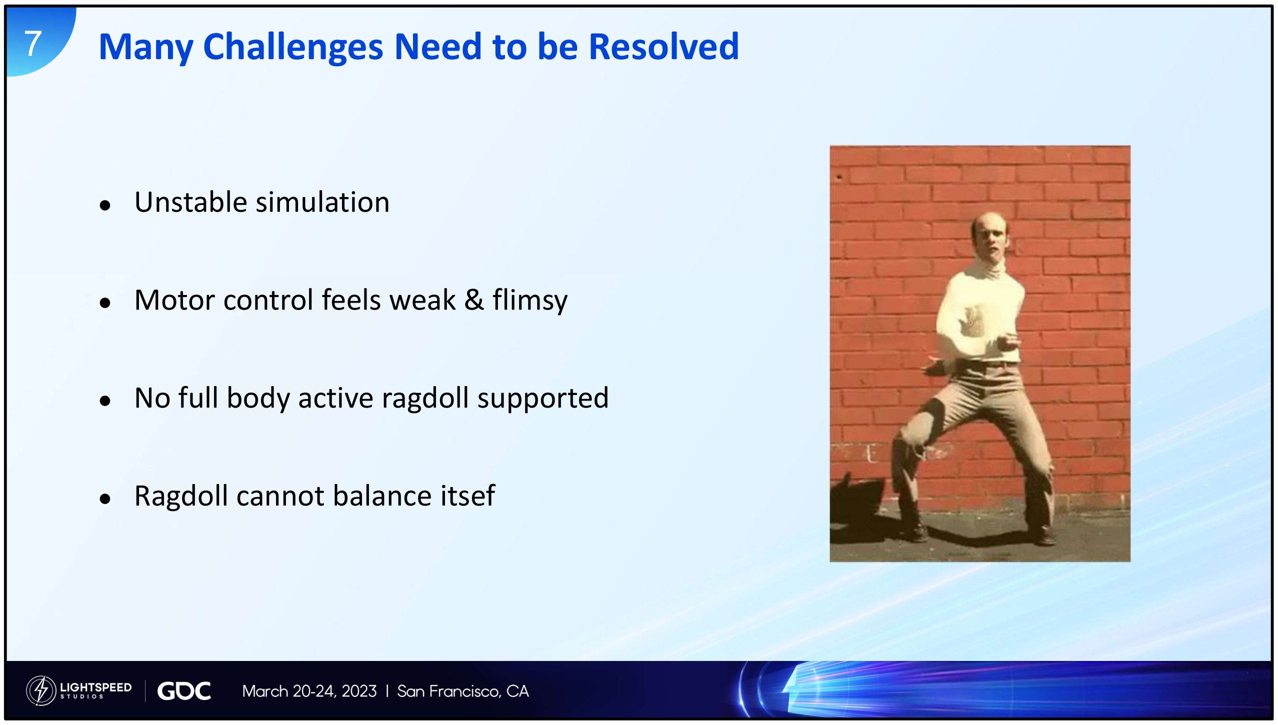
6

This is Where We Started With

LIGHTSPEED STUDIOS | GDC March 20-24, 2023 I San Francisco, CA

This video demonstrates the physical animation system offered by the Unreal Engine. As you can see, when the simulation is activated, the body parts appear to be somewhat out of control, especially when they receive impacts. In some extreme cases, the body parts briefly become disconnected from each other.

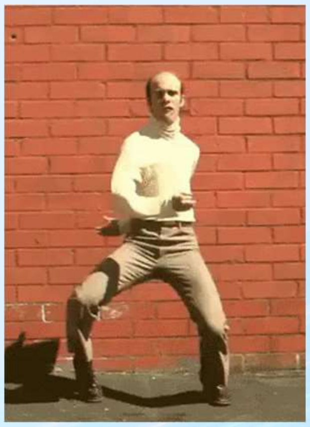
Another issue is that the character's root must always be anchored to the world coordinate, or else it will collapse.



7

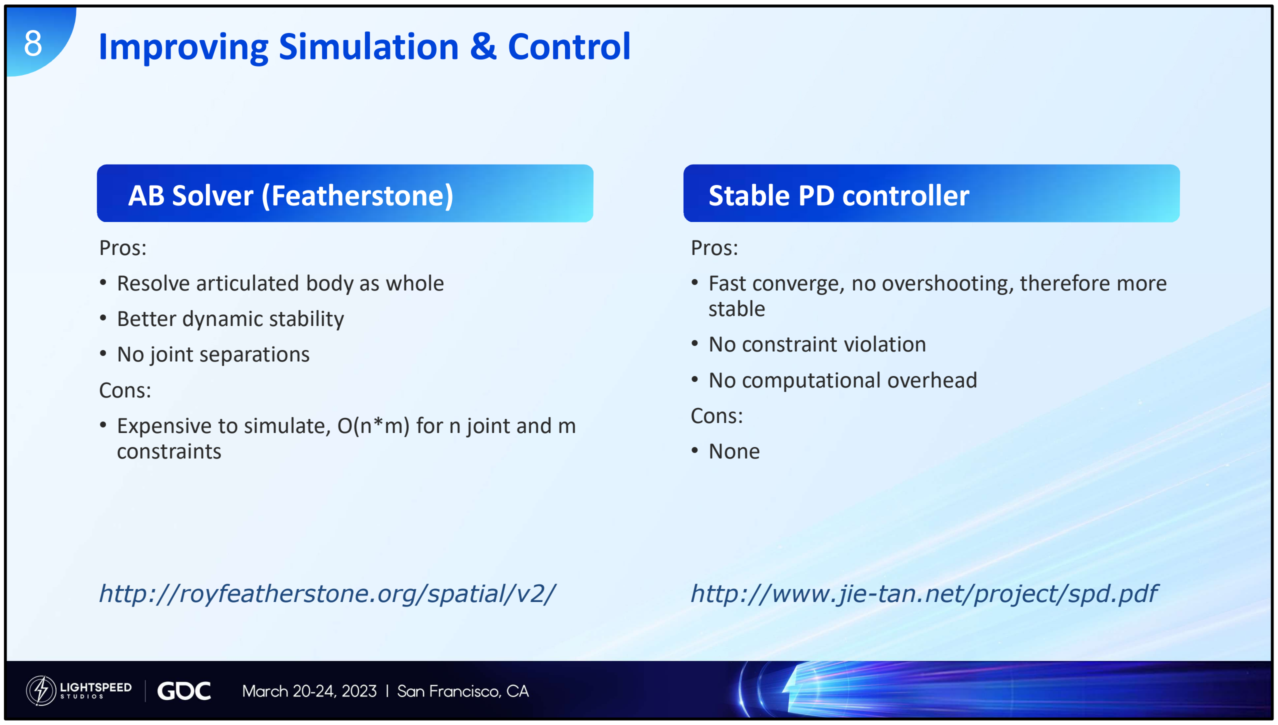
Many Challenges Need to be Resolved

- Unstable simulation
- Motor control feels weak & flimsy
- No full body active ragdoll supported
- Ragdoll cannot balance itsef



LIGHTSPEED STUDIOS | GDC March 20-24, 2023 | San Francisco, CA

Here's a summary of the challenges that need to be resolved. Most of these issues have already been covered in the previous video. The question is: where do we get started?


8

Improving Simulation & Control

AB Solver (Featherstone)

Pros:

- Resolve articulated body as whole
- Better dynamic stability
- No joint separations

Cons:

- Expensive to simulate, O(n*m) for n joint and m constraints

http://royfeatherstone.org/spatial/v2/

Stable PD controller

Pros:

- Fast converge, no overshooting, therefore more stable
- No constraint violation
- No computational overhead

Cons:

- None

http://www.jie-tan.net/project/spd.pdf

LIGHTSPEED STUDIOS | GDC March 20-24, 2023 I San Francisco, CA

We started by improving the simulation and controls, which are the fundamentals of the physical animation system.

We implemented Featherstone's articulated body algorithm, called ABSolver, for ragdoll simulation

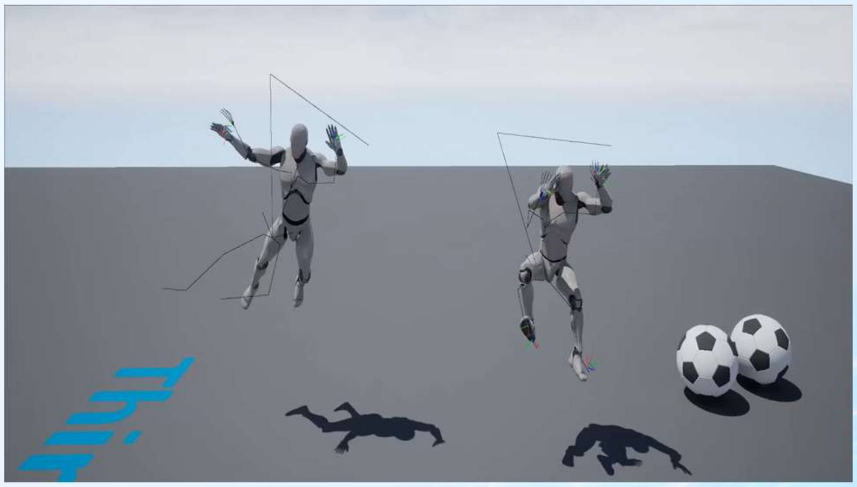
And built a stable PD controller for ragdoll controlling.

The main difference between ABSolver and conventional constraint solvers is that the ABSolver solves articulated bodies as whole, rather than as set of independent constraints. The result is much more stable and accurate. Though it does come with additional performance cost.

The stable PD controller, on the other hand, computes the control forces using the predicted state of the next time step, thus converging much faster than the traditional PD controller. Furthermore, it can be seamlessly integrated into the AB Solver with no additional performance overhead.

9

Now, the Movement Looks Much Better



LIGHTSPEED STUDIOS | GDC March 20-24, 2023 | San Francisco, CA

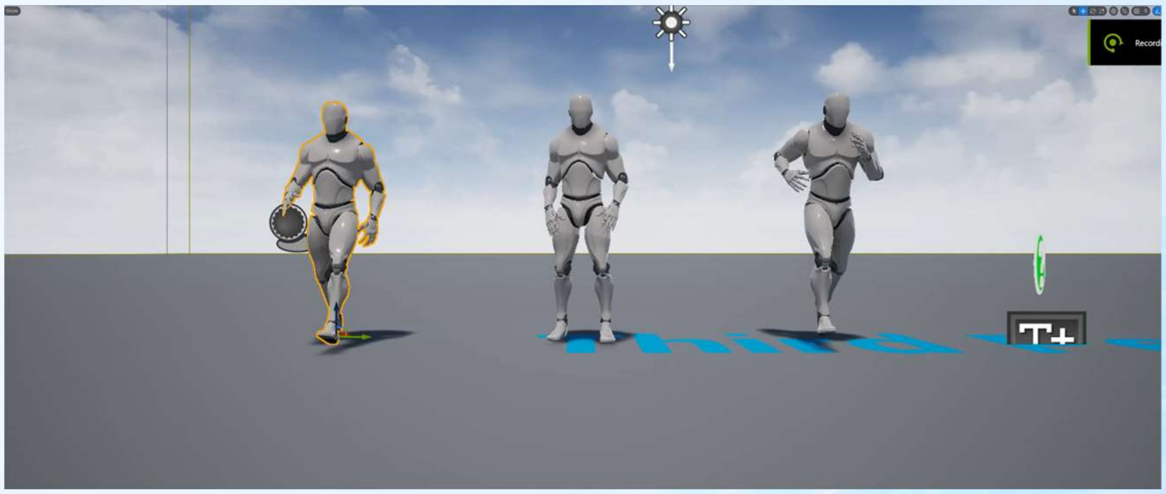
Let's take a look at how the Ragdoll movement performs with the new simulation and control schemes. It looks much smoother and natural, doesn't it?

The black lines show the underlined target animation.

The impact reaction looks incredibly realistic as well.

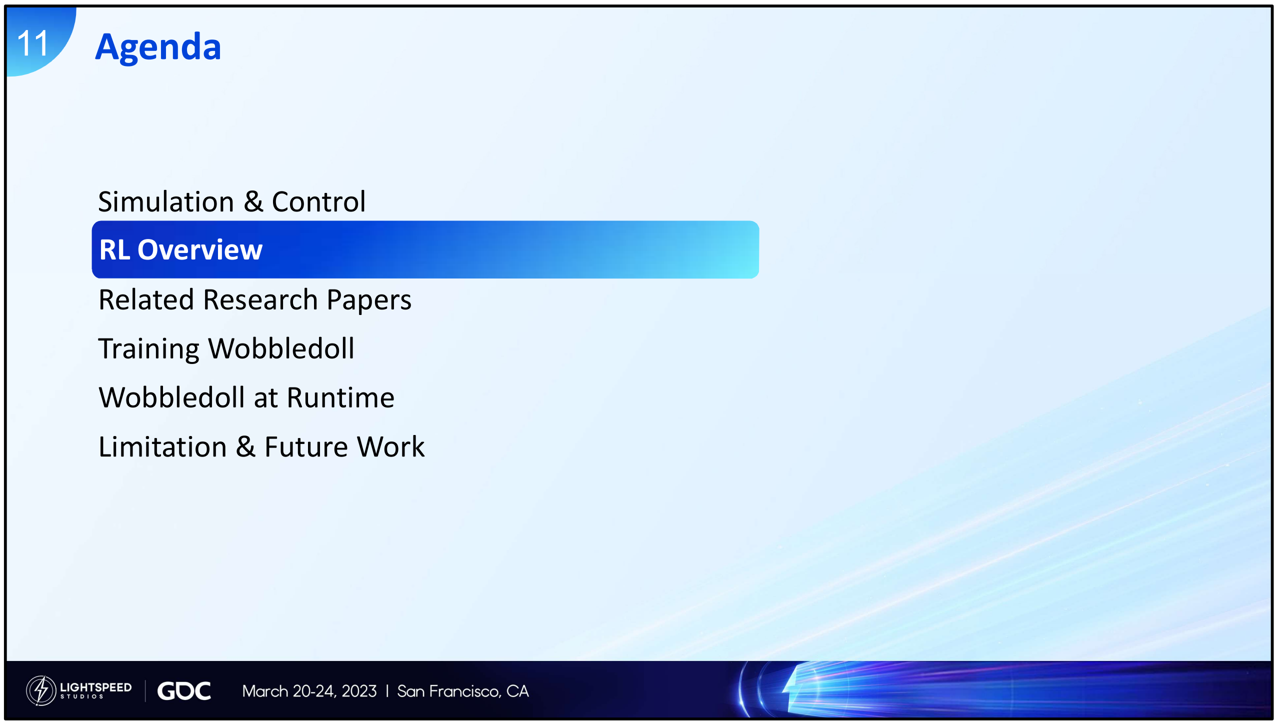
10

Well, When Placing Them on the Ground...



Ok, what if we place these characters on the ground?

This is because they have no idea how to maintain the balance.



11

Agenda

Simulation & Control

RL Overview

Related Research Papers

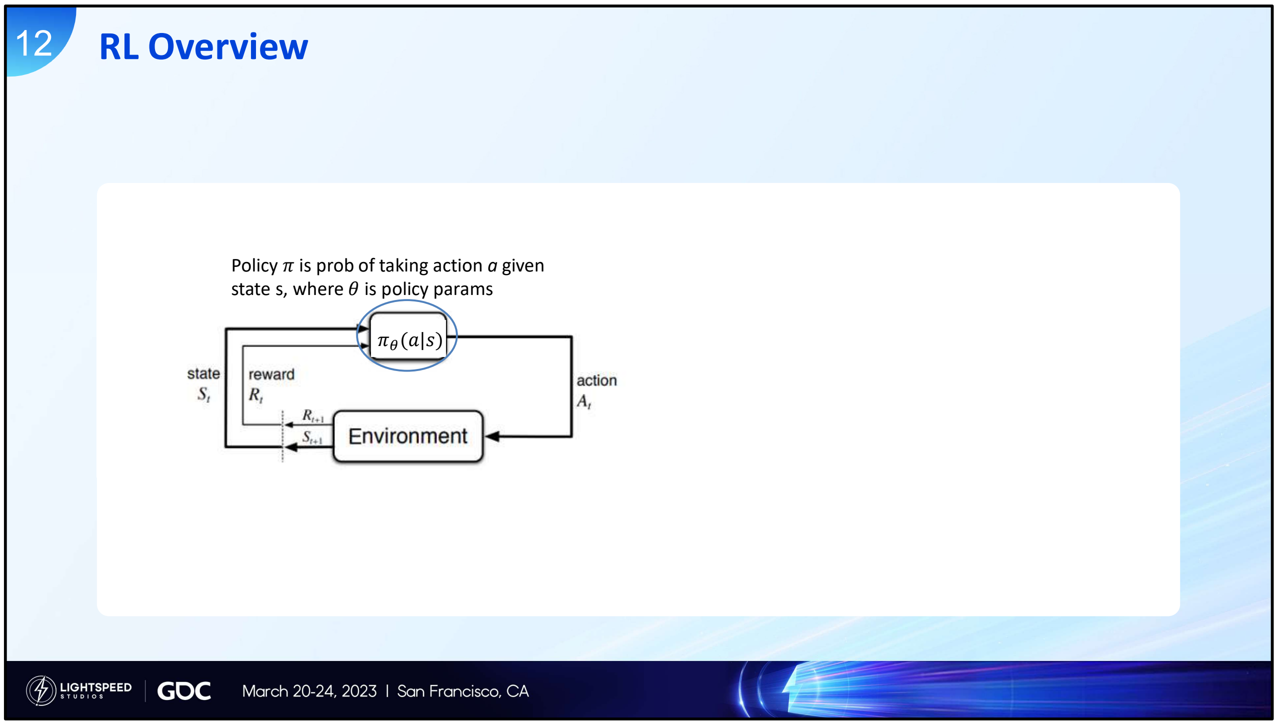
Training Wobbledoll

Wobbledoll at Runtime

Limitation & Future Work

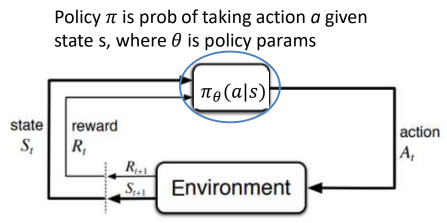
Alright, before diving into the complexities of ragdoll balancing, we need to have some basic understanding of reinforcement learning.

Let’s start with covering some basics.



12

RL Overview



Policy π is prob of taking action a given state s, where θ is policy params
$\pi_\theta(a|s)$
state S_t
reward R_t
R_{t+1}
S_{t+1}
Environment
action A_t

LIGHTSPEED STUDIOS | GDC March 20-24, 2023 I San Francisco, CA

This is the basic reinforcement learning process, the Markov Decision Process, which describes how the basic workflow looks like.

It consists of a few key elements:

## Agent is a policy network that we are training to make good decisions. We use π to represent the policy, and θ to represent policy parameters

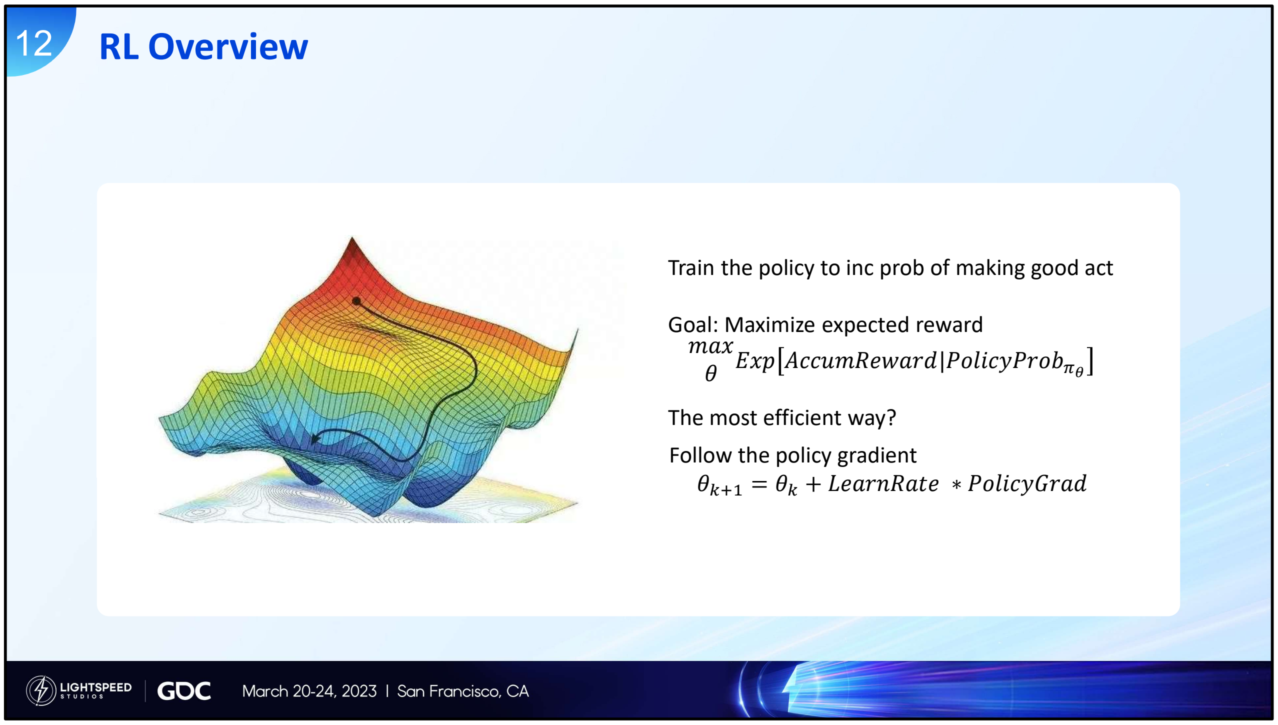
## Environment is a dynamic model that agent interacts with.

## States define the current situation of the agent within

the environment, which used by the agent as input.

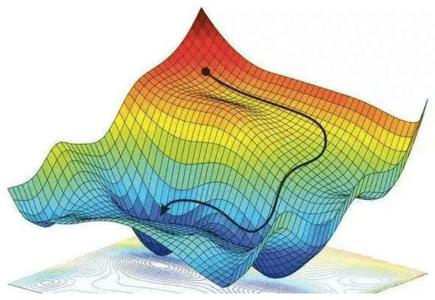
## Action is the decision an agent makes at given states and policy, which will change how the agent interacts with the environment.

## Rewards evaluate how well the agent performs, and they are used to train the policy network.



12

RL Overview



Train the policy to inc prob of making good act

Goal: Maximize expected reward

$$\max_{\theta} Exp[AccumReward|PolicyProb_{\pi_\theta}]$$

The most efficient way?

Follow the policy gradient

$$\theta_{k+1} = \theta_k + LearnRate * PolicyGrad$$

LIGHTSPEED STUDIOS | GDC March 20-24, 2023 | San Francisco, CA

## Basically, the whole point of RL is to train the policy to increase the probability of making good actions.

This can be represented as maximizing the expected reward, which is the weighted average of sum of rewards for all possible actions that policy π_θ can produce.

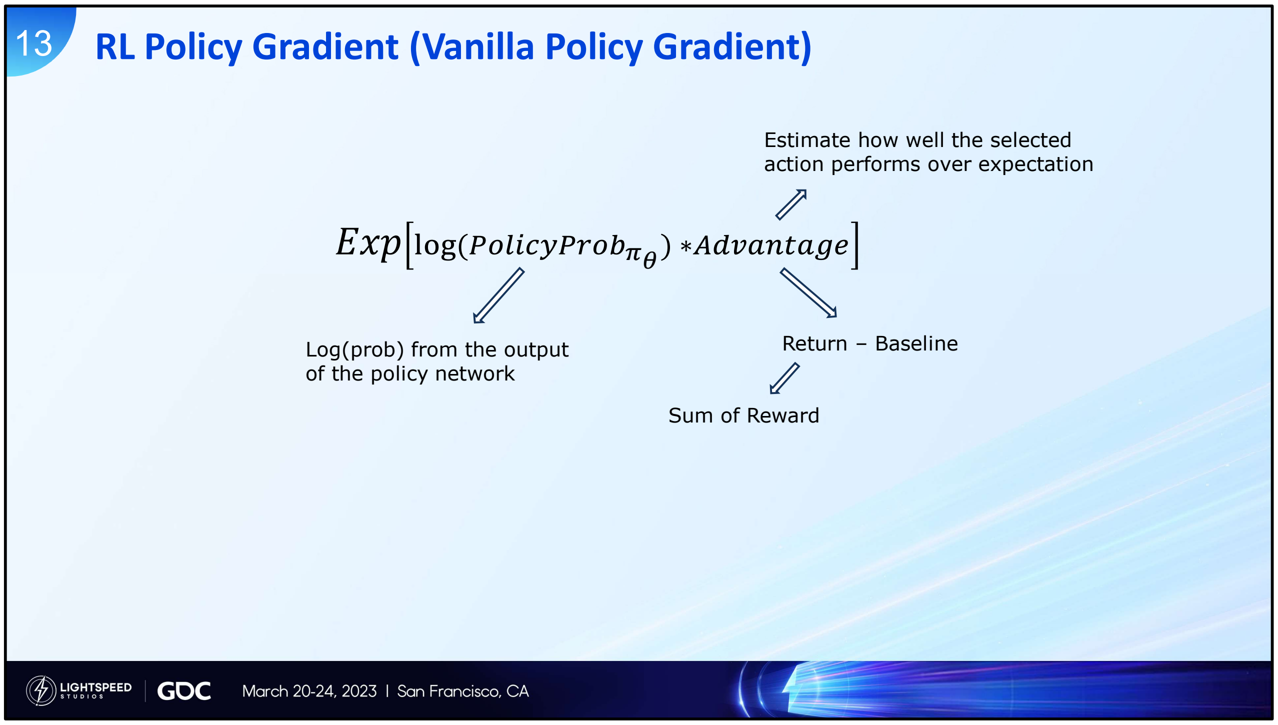
## But what is the most efficient way of doing this?

Let’s look at the picture on the left, which plots a cost function of policy in 3d space. The cost function basically measures how close a policy is match to its objective.

So we want to minimize the cost, means traverse down the surface. The most efficient way would be going along with the steepest slop, which called policy gradient.

## and here we go, we update the policy parameters by baby stepping along the gradient direction, which is learning rate times policy gradient.

Ok, what is the policy gradient look like?


13 RL Policy Gradient (Vanilla Policy Gradient)

$$Exp\left[\log(PolicyProb_{\pi_\theta}) * Advantage\right]$$

Estimate how well the selected action performs over expectation

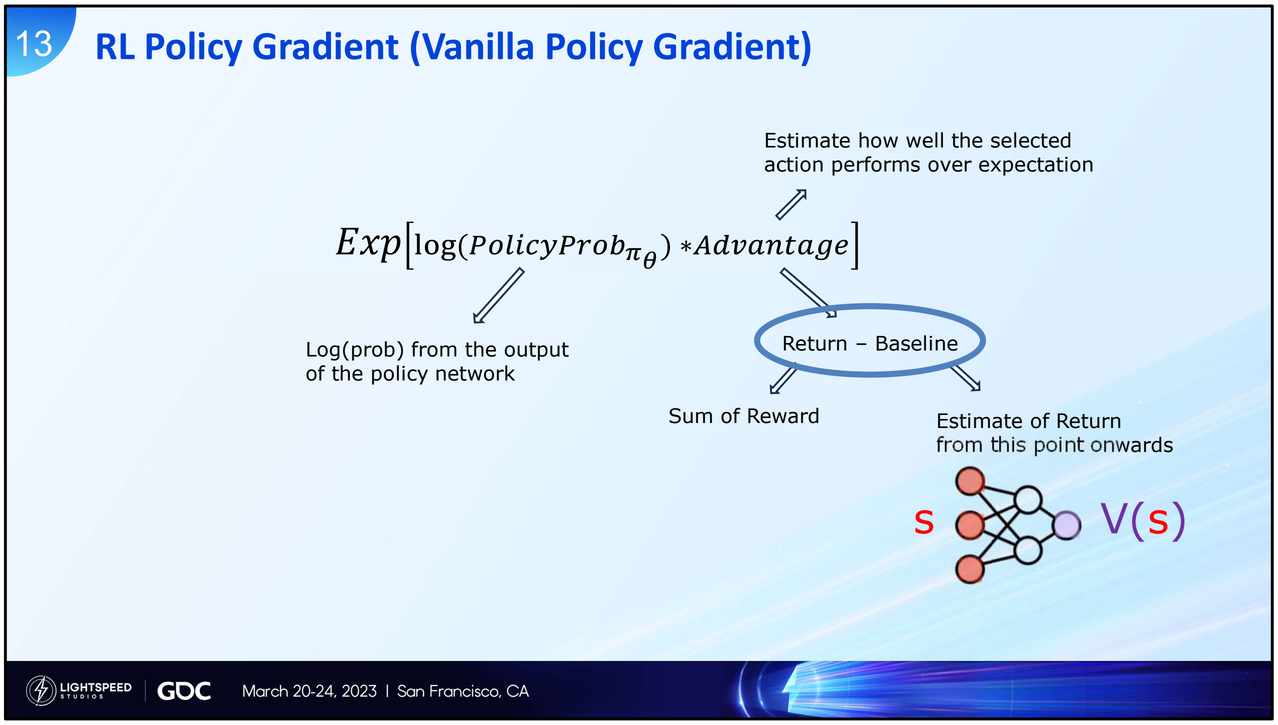
Log(prob) from the output of the policy network

Return – Baseline

Sum of Reward

There are actually several different ways of computing policy gradient, here we use Vanilla Policy Gradient for simplicity
It is defined as the expectation over the log of the policy actions times an estimate of the advantage function.
Ok, so what is that all means
## The first term is our policy. It is log of the probability of taking an action from the policy network of given state.
## The second term is the advantage function, which basically tries to estimate the relative value of the selected action in the current state over the expectation.
## In order to compute the advantage, we need two things, we need sum of reward, which also called return, and we need a baseline estimate.
## The sum of reward is the sum of all the rewards that the agent generates from current state and action to the end of episode. Such

reward usually comes with discount factor over time, here we ignore that for simplicity

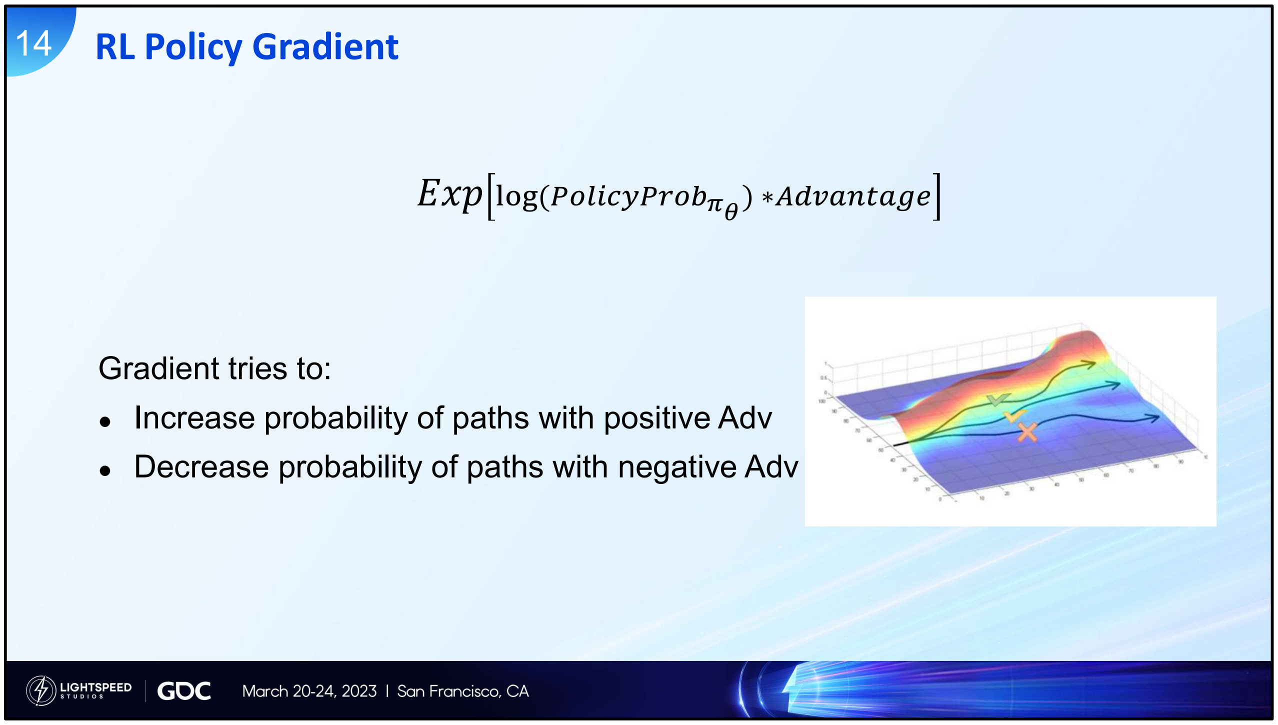


RL Policy Gradient (Vanilla Policy Gradient)

13
Estimate how well the selected action performs over expectation
$Exp\left[\log(PolicyProb_{\pi_\theta}) * Advantage\right]$
Log(prob) from the output of the policy network
Return – Baseline
Sum of Reward
Estimate of Return from this point onwards
s
V(s)
LIGHTSPEED STUDIOS
GDC
March 20-24, 2023 | San Francisco, CA

## And then the second part is the baseline or the value function. What it does is trying to give an estimate of sum of reward from current state and onward.
Basically it’s trying to guess what the final return is by the end of episode starting from the current state.
## During the training, this function is going to be frequently updated using the collected experiences. So it can be trained as a separate neural network along the way by supervised learning
What the supervised learn is, is basically training the policy with labeled data and try to make the policy output matches with labeled output. Due to the time limit, we are not going into the details of it. For now, we can think it as a function that estimates the expected return at given state.
##So now, we have the actual return from collected samples, then subtract the expected return that estimated from value network.

This tells us if the current policy is over performed or under performed to our expectation.
This is basically what advantage function does.
Now we have basic understanding of each piece of policy gradient function, but how does it actually work?



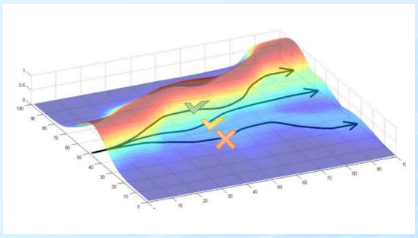
14

RL Policy Gradient

$$Exp\left[\log(PolicyProb_{\pi_\theta}) * Advantage\right]$$

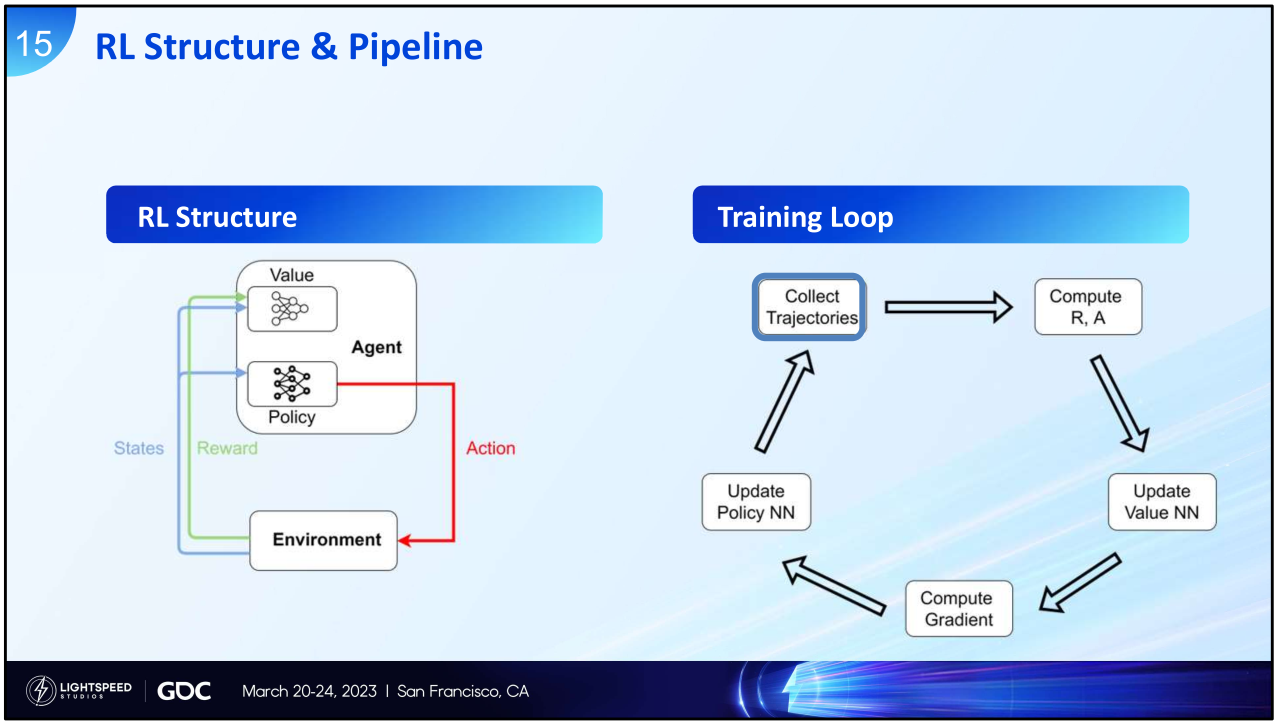
Gradient tries to:

- Increase probability of paths with positive Adv
- Decrease probability of paths with negative Adv



LIGHTSPEED STUDIOS | GDC March 20-24, 2023 I San Francisco, CA

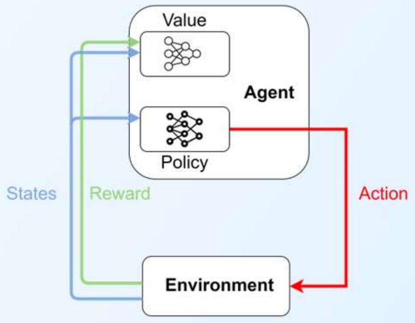
It's actually surprisingly simple. If the actions that the agent took in the sampled path resulted in better than average return, then what we'll do is increase the probability of selecting them again.
Vise vera, if the advantage function was negative, then we will reduce the likelihood of the selecting those actions.
Overtime, the agent will always take good actions and avoid bad ones.



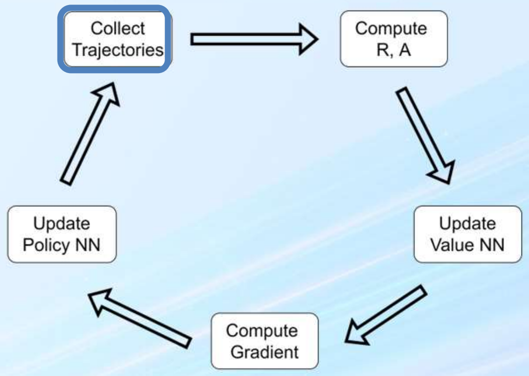
15

RL Structure & Pipeline

RL Structure

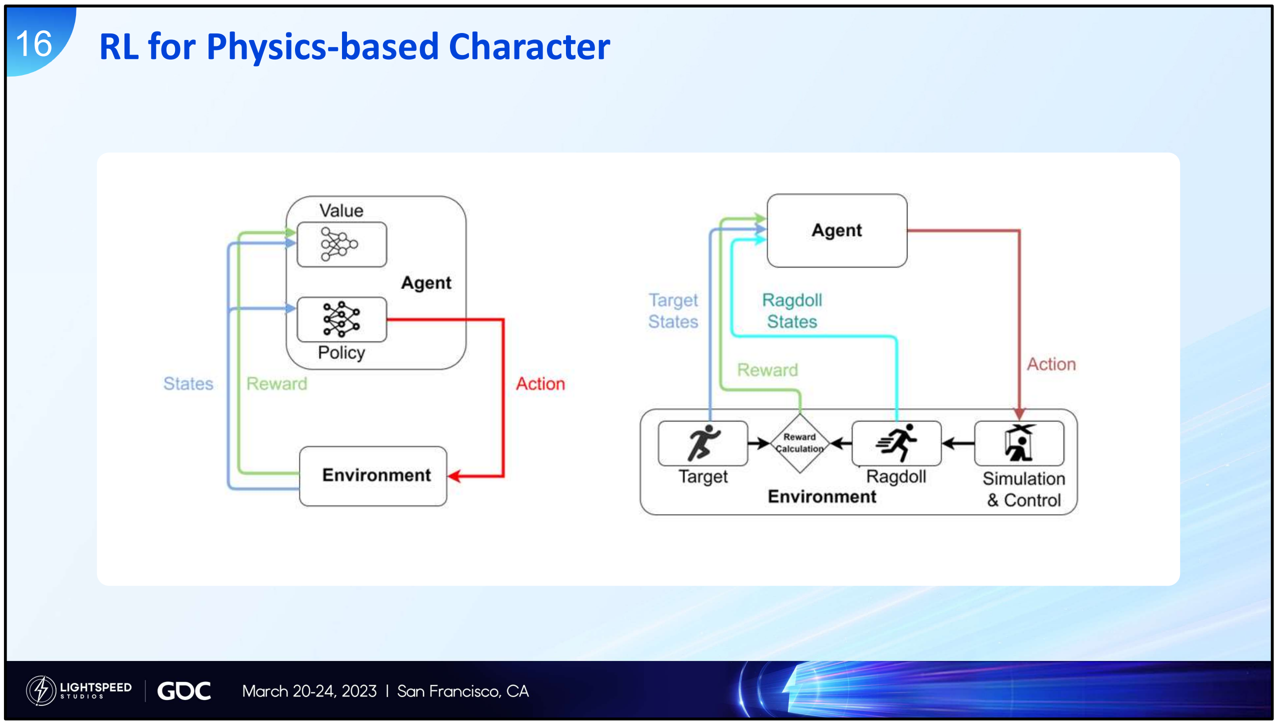


Training Loop



LIGHTSPEED STUDIOS | GDC March 20-24, 2023 I San Francisco, CA

Now, let's look at what the RL structure and training loop would be. As we mentioned earlier, we are training the value network along the side. So now we have both networks in our agent.
Then the update order would be,
##collecting the samples,
##compute return and advantage,
##and use that to update the Value network
##Then compute the gradient,
##and use that to update the policy network.
##And then we start over with collecting sample with our new policy again. So on and so forth.



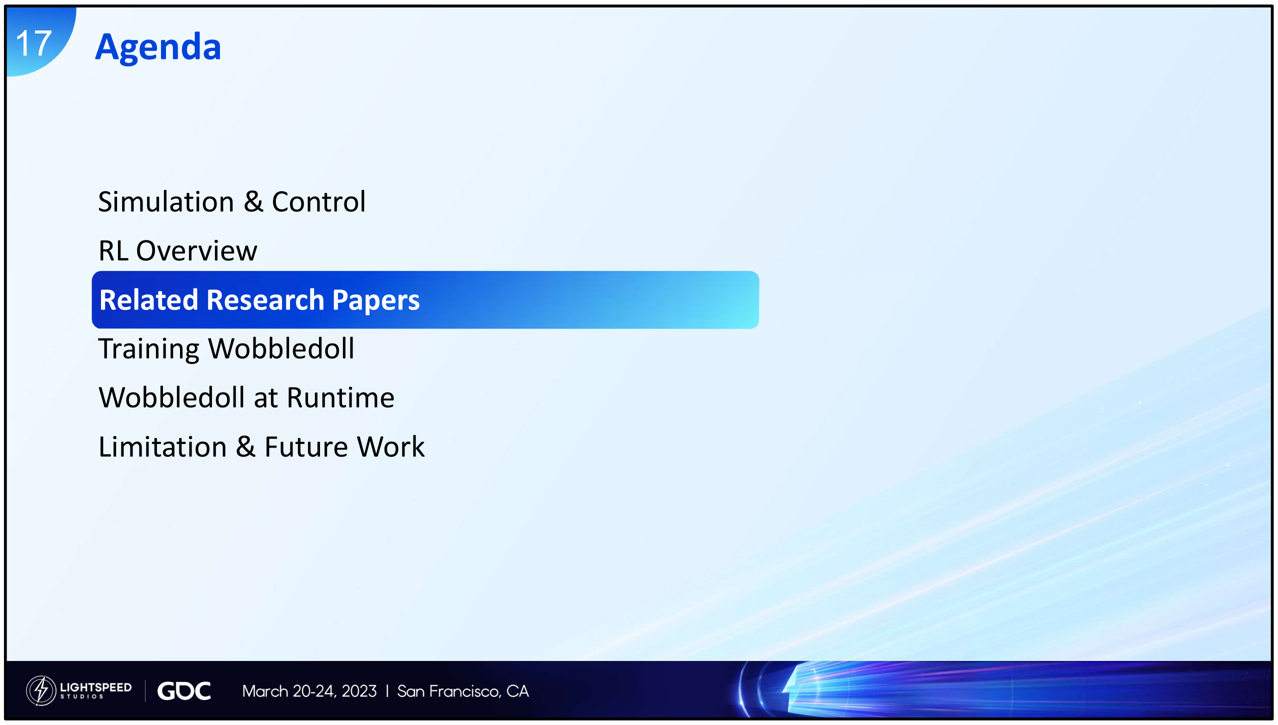
16

RL for Physics-based Character

LIGHTSPEED STUDIOS | GDC March 20-24, 2023 I San Francisco, CA

Lastly, let’s take a quick look how RL can be applied to the physics-based animation.
Start with basic physical animation update, which we have a target motion, then the physics update to drive the ragdoll to follow the target, which generates the ragdoll motion.
## We want to train an agent to help controlling the ragdoll. To do so, we pass in both target states and ragdoll states to the agent, so it fully aware of the current motion and target motion. The agent will then generate action from its policy network, which will be applied as some form of control signals to manipulate the ragdoll. So we will get an updated ragdoll motion after physics update.
## The final part of the process is that we need to evaluate how well the agent performs and come up a reward to train the agent. There are many different designs for reward calculations, but usually it involves comparing ragdoll with target motion to find out how well

they match.



17

Agenda

Simulation & Control

RL Overview

Related Research Papers

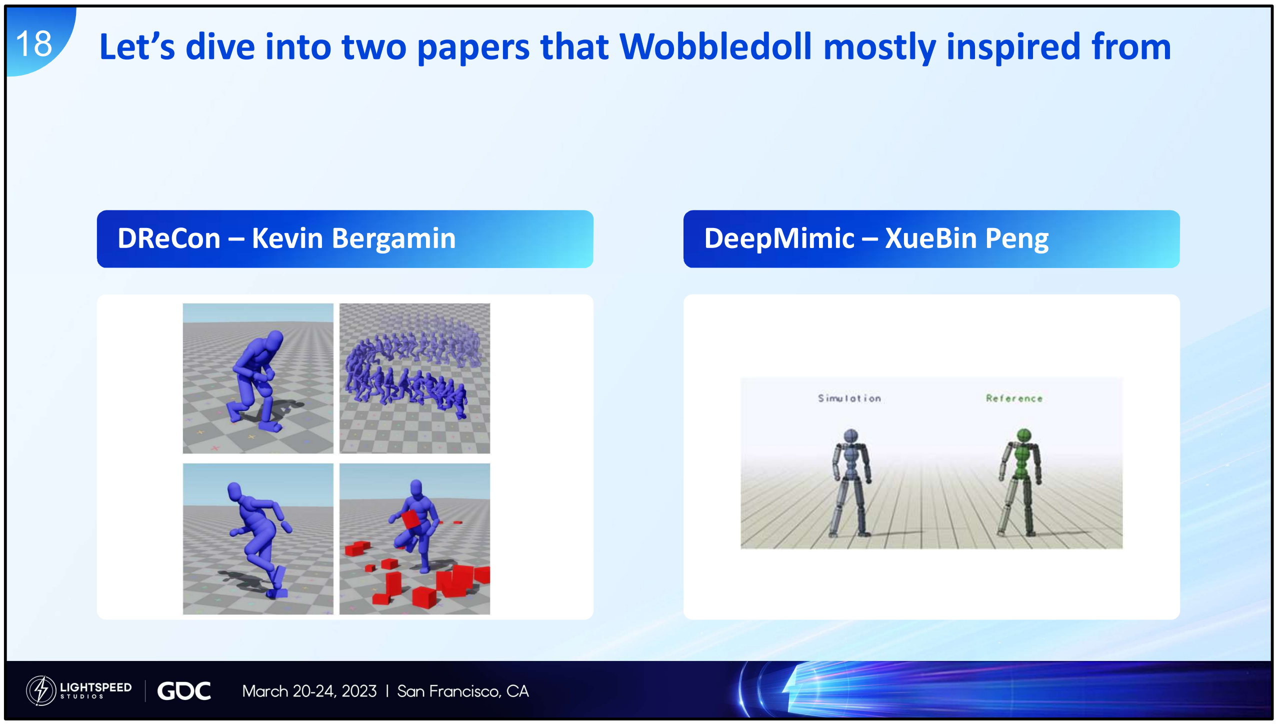
Training Wobbledoll

Wobbledoll at Runtime

Limitation & Future Work

LIGHTSPEED STUDIOS | GDC March 20-24, 2023 | San Francisco, CA

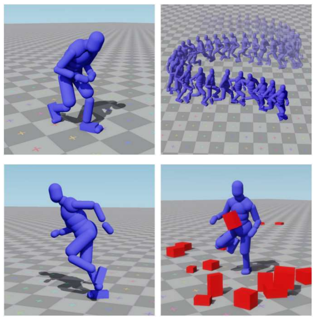
Alright, now since we know the basics of RL. let’s dive into a few research papers that Wobbledoll implementation is based on.



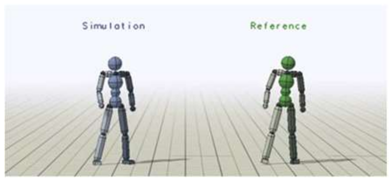
18

Let’s dive into two papers that Wobbledoll mostly inspired from

DReCon – Kevin Bergamin

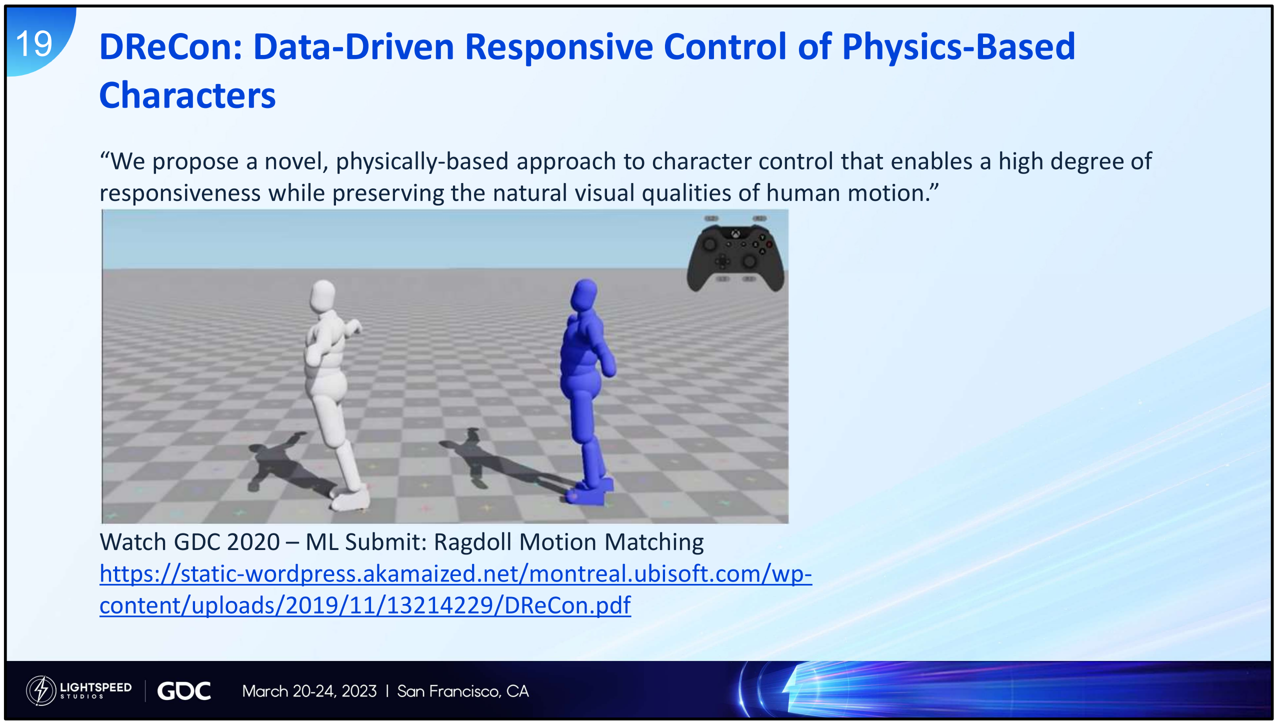


DeepMimic – XueBin Peng



LIGHTSPEED STUDIOS | GDC March 20-24, 2023 I San Francisco, CA

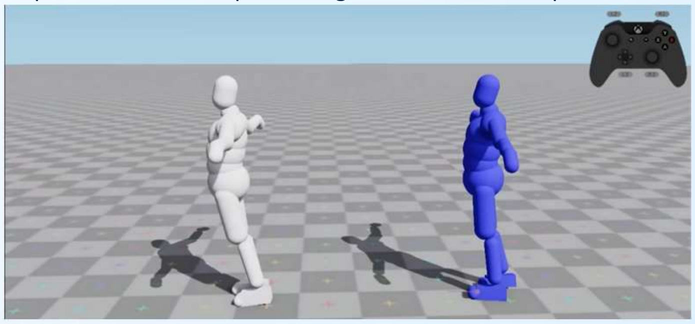
Wobbledoll system was inspired and borrow ideas from quite many papers. Due to the time limit, I will only go through two of them here, DReCon and DeepMimic



19

DReCon: Data-Driven Responsive Control of Physics-Based Characters

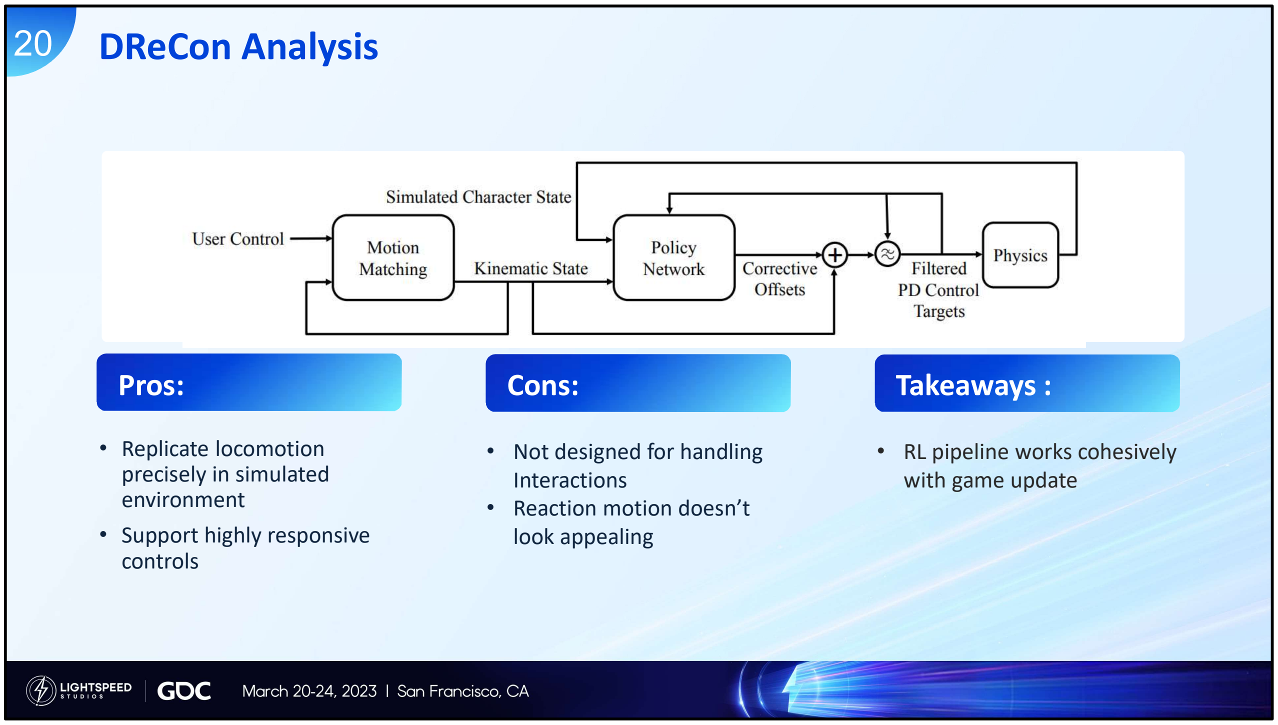
"We propose a novel, physically-based approach to character control that enables a high degree of responsiveness while preserving the natural visual qualities of human motion."



Watch GDC 2020 – ML Submit: Ragdoll Motion Matching
https://static-wordpress.akamaized.net/montreal.ubisoft.com/wp-content/uploads/2019/11/13214229/DReCon.pdf

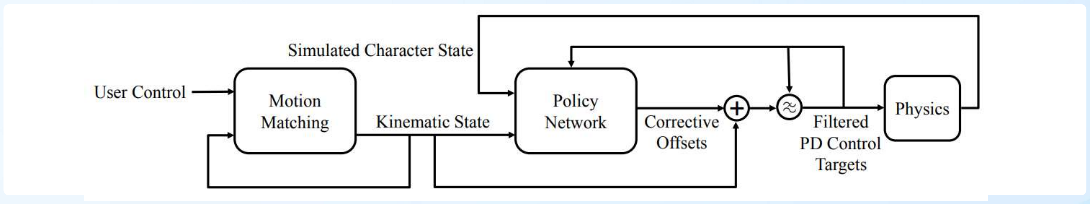
LIGHTSPEED STUDIOS | GDC March 20-24, 2023 | San Francisco, CA

Let's start with DReCon. DReCon is a very cool algorithm, in which it manages to train the ragdoll to perform locomotion in simulated environment.
As you see in this video, the white guy runs locomotion, and blue guy runs simulation. You can barely tell the difference. The motion is precisely matched and the control is very responsive.
There is talk in GDC 2020 introduces this algorithm in depth. Definitely worth to check it out if you are interested.



20

DReCon Analysis



Pros:

- Replicate locomotion precisely in simulated environment
- Support highly responsive controls

Cons:

- Not designed for handling Interactions
- Reaction motion doesn't look appealing

Takeaways :

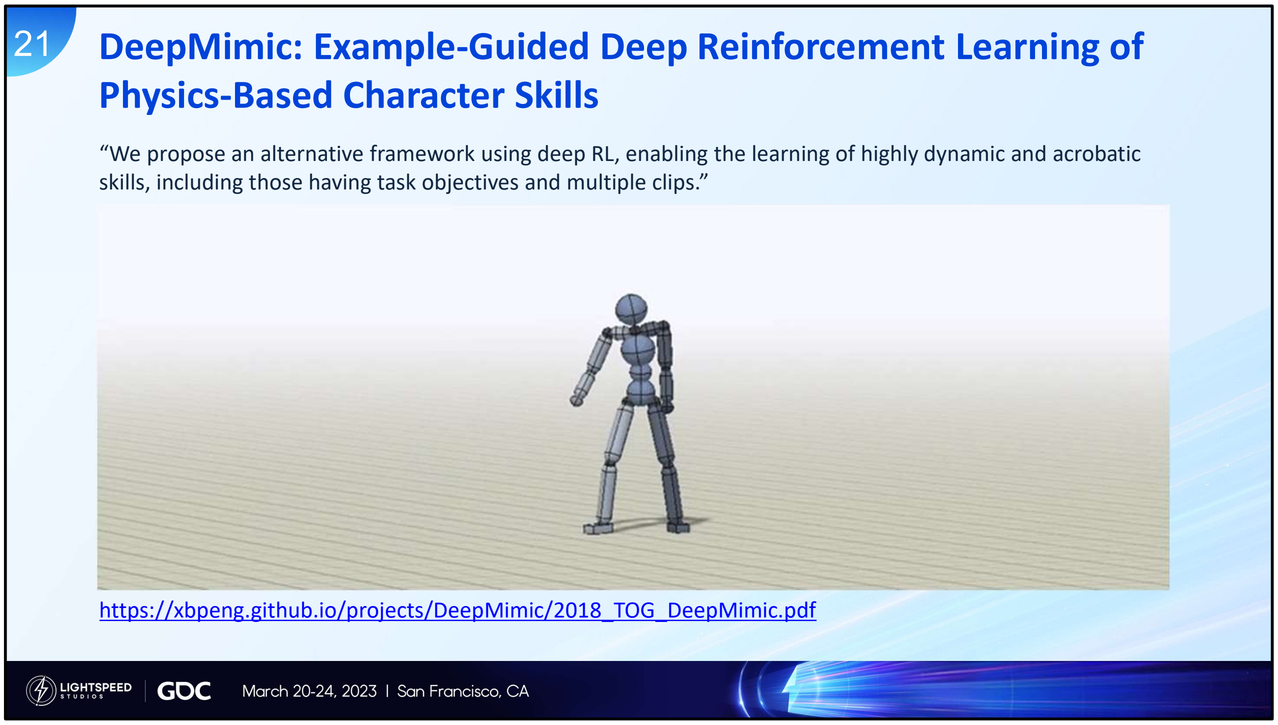
- RL pipeline works cohesively with game update

LIGHTSPEED STUDIOS | GDC March 20-24, 2023 I San Francisco, CA

Let's look at the DReCon update pipeline. If you remember the structure we mentioned in earlier tutorial, this is actually very similar. The motion matching is basically an advanced locomotion system. It uses that to generate target state. The policy also takes ragdoll state from simulation. The output actions are the corrective offsets, which gets applied to the target motion before passing into the simulation & control.
Reward function wise, DReCon uses multiple sub rewards to make sure the ragdoll poses, motion and velocities are matching to the target.

This pipeline works cohesively with game update, especially that it works directly with the locomotion system. This is the main idea we borrowed for the Wobbledoll implementation.

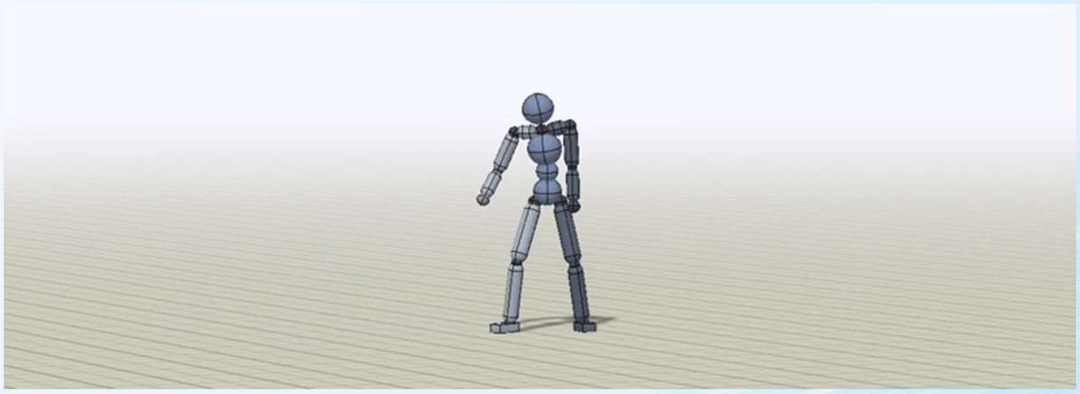
The DReCon was designed to replicate locomotion. So, it has some limitations when handling impacts and physical interactions. I will discuss later how Wobbledoll improves on those asepcts.



21

DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills

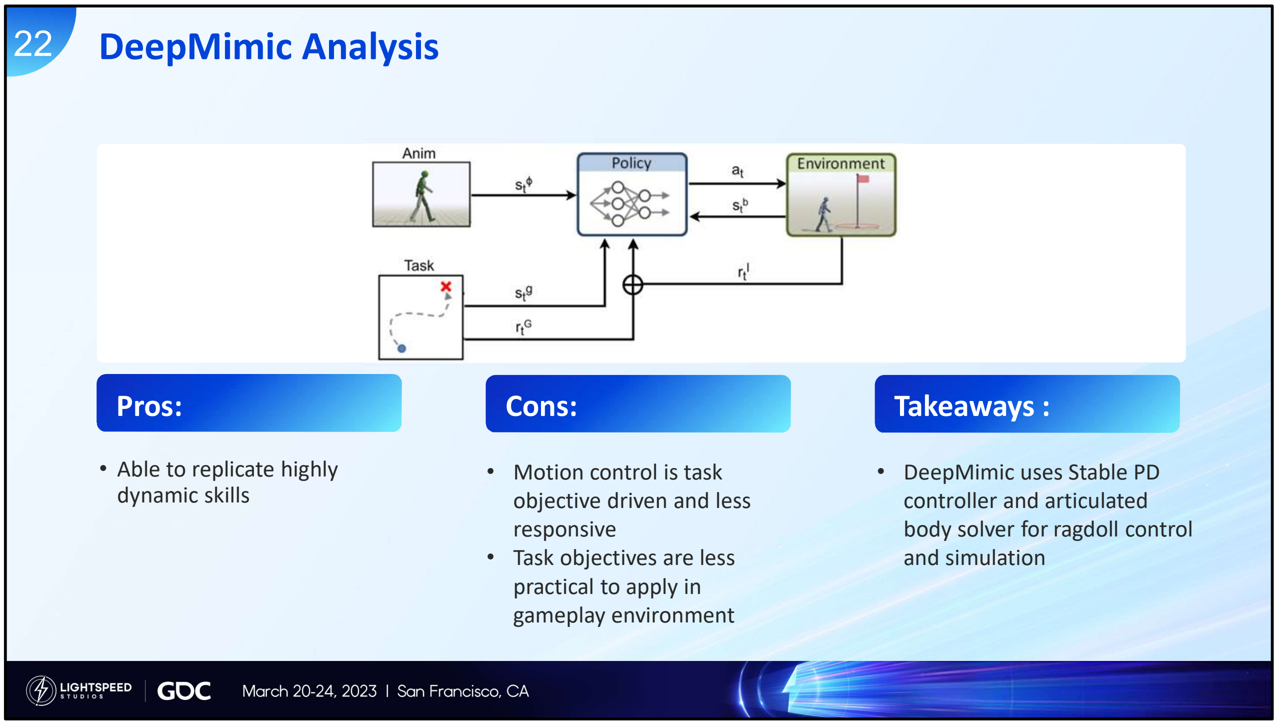
"We propose an alternative framework using deep RL, enabling the learning of highly dynamic and acrobatic skills, including those having task objectives and multiple clips."



https://xbpeng.github.io/projects/DeepMimic/2018_TOG_DeepMimic.pdf

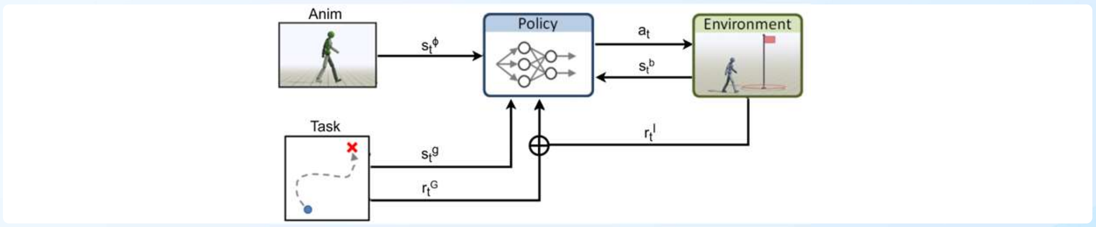
LIGHTSPEED STUDIOS | GDC March 20-24, 2023 I San Francisco, CA

Next, I'd like to talk about DeepMimic. DeepMimic is likely the first paper to explore training a ragdoll agent to accurately imitate a target motion. Additionally, the trained ragdoll can also carry out assigned tasks, such as walking to a specific destination or kicking a particular target.



22

DeepMimic Analysis



Pros:

- Able to replicate highly dynamic skills

Cons:

- Motion control is task objective driven and less responsive
- Task objectives are less practical to apply in gameplay environment

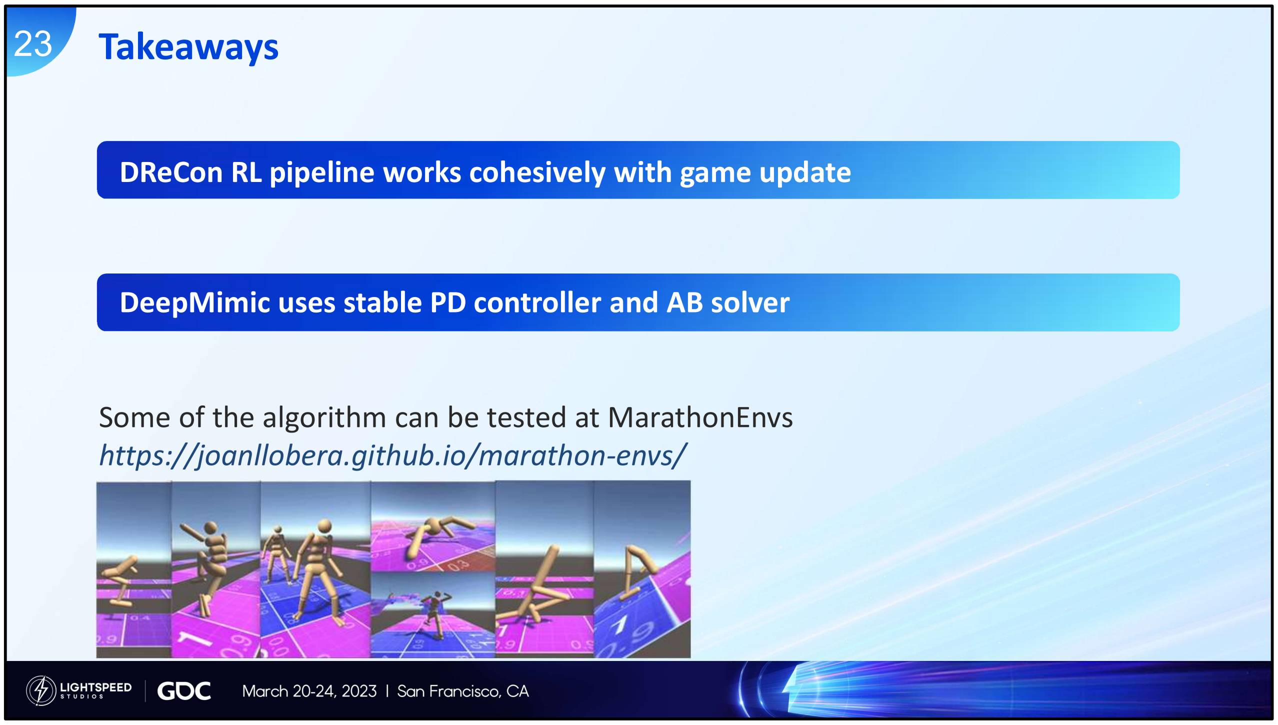
Takeaways :

- DeepMimic uses Stable PD controller and articulated body solver for ragdoll control and simulation

LIGHTSPEED STUDIOS | GDC March 20-24, 2023 I San Francisco, CA

Structure wise, the key difference here is that the DeepMimic takes task as part of the states and part of the reward calculation. That's the reason why the trained policy is able to complete a given task while replicate target motion. However, the task driven scheme doesn't work too well with gameplay environment as games require more responsive control and interactions.

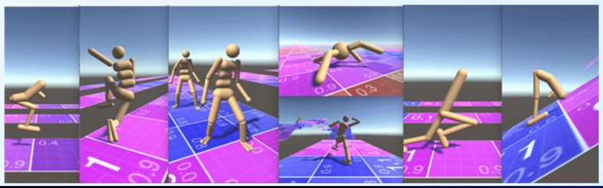
Interestingly, the DeepMimic environment uses PD controller and articulated body solver for ragdoll control and simulation, which are the same as what we have. So our takeaway here is how DeepMimic agent applying actions to the simulation environment.


23 Takeaways

DReCon RL pipeline works cohesively with game update

DeepMimic uses stable PD controller and AB solver

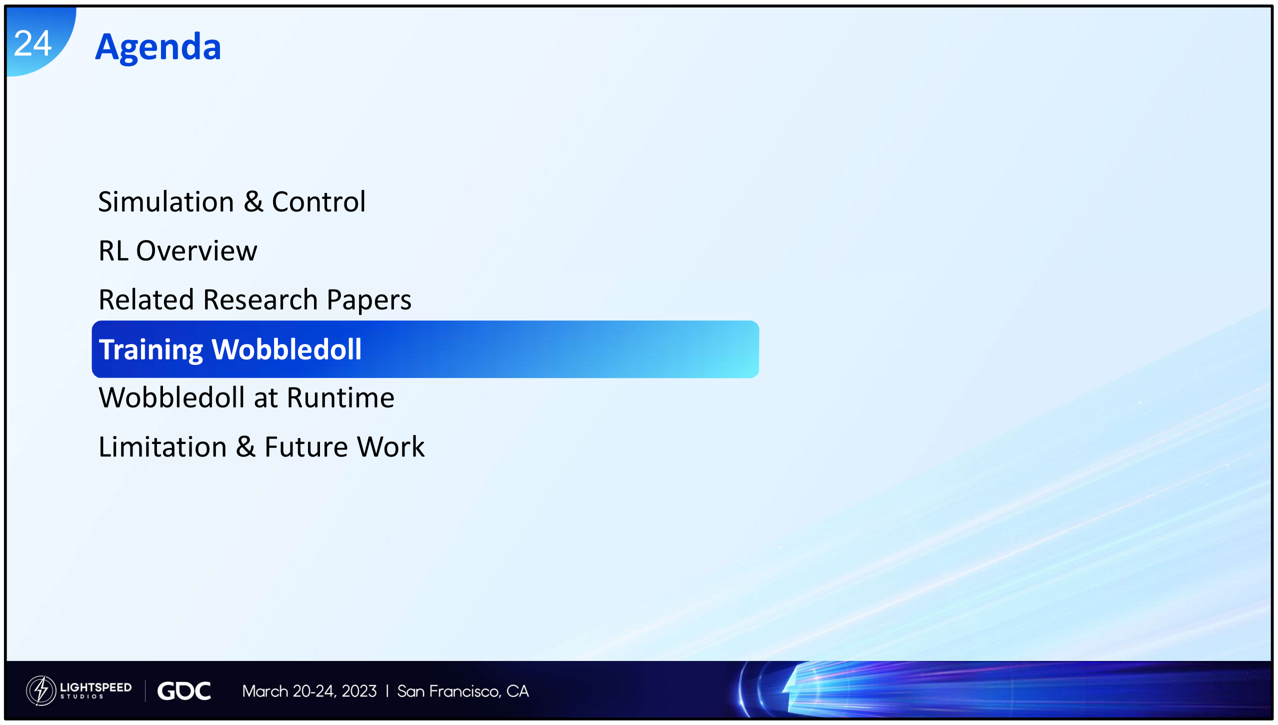
Some of the algorithm can be tested at MarathonEnvs
https://joanllobera.github.io/marathon-envs/



LIGHTSPEED STUDIOS | GDC March 20-24, 2023 I San Francisco, CA

Now’s let’s sum up the key takeaways from 2 papers we analyzed.
DReCon offers great RL pipeline that works cohesively with the game update
DeepMimic uses same control and simulation scheme as our game environment, so we can follow the way how its agent applies action

##As a side note, I'd like to advertise an open-source project called MarathonEnvs. Some of the papers I've talked about here have already been implemented in this project. MarathonEnv is built upon the Unity engine and is very easy to use. If you're looking to get some hands-on experience with RL-powered ragdolls, this is the go-to project.



24

Agenda

Simulation & Control

RL Overview

Related Research Papers

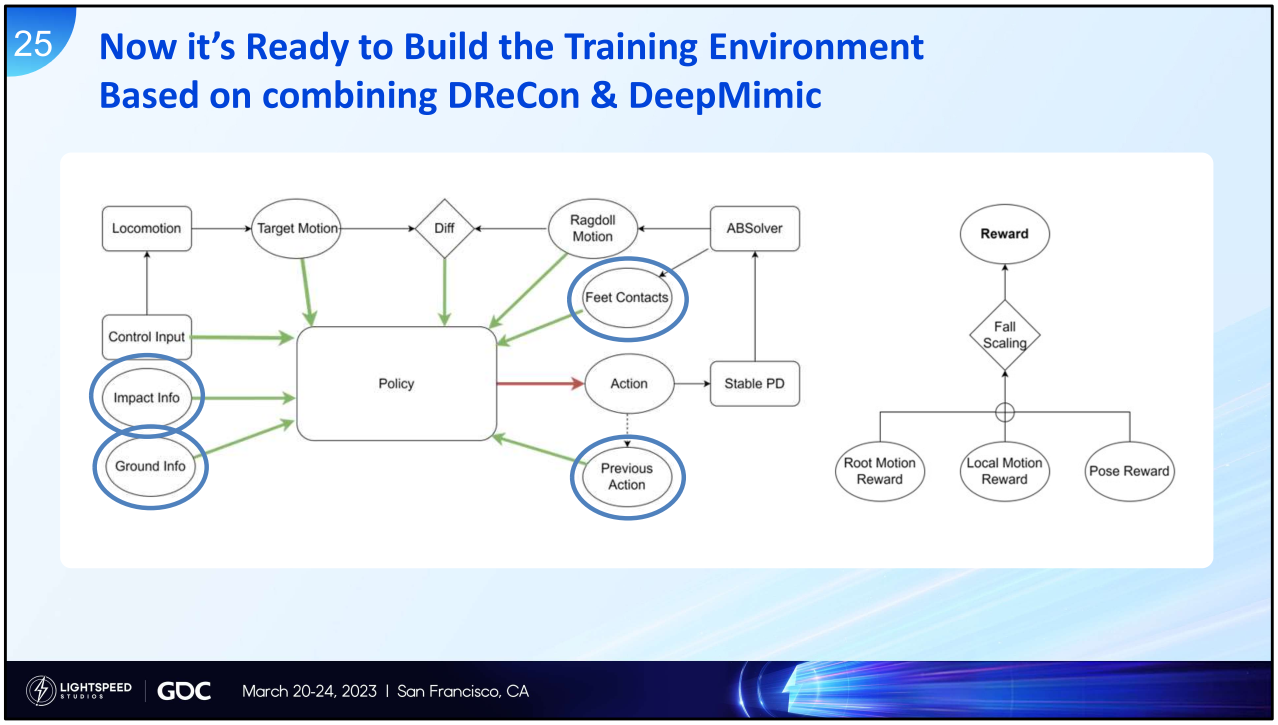
Training Wobbledoll

Wobbledoll at Runtime

Limitation & Future Work

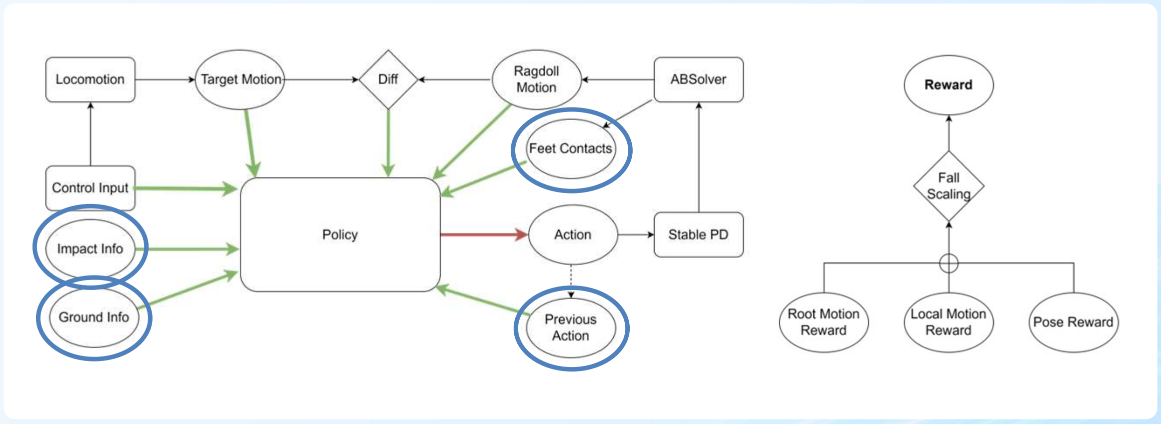
LIGHTSPEED STUDIOS | GDC March 20-24, 2023 | San Francisco, CA

Alright, now it's time to talk about how we train the Wobbledoll



25

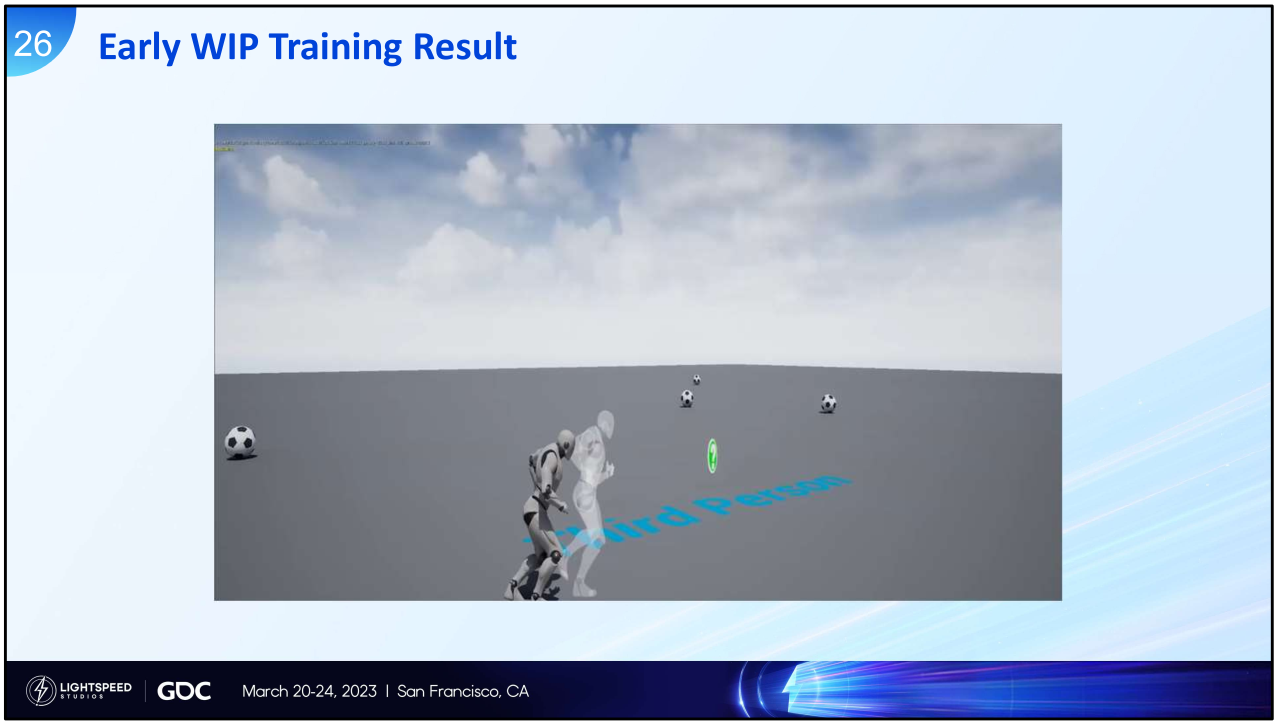
Now it's Ready to Build the Training Environment
Based on combining DReCon & DeepMimic



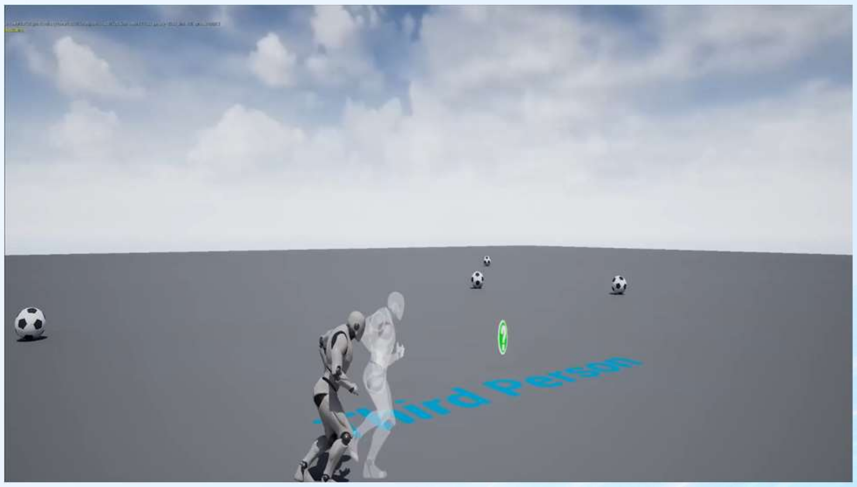
LIGHTSPEED STUDIOS | GDC March 20-24, 2023 I San Francisco, CA

Recap the research takeaways we talked about earlier. We are going to follow the DReCon update pipeline and apply the action the same way as DeepMimic does.
So we feed the agent the target motion and ragdoll motion as input states, then it would generate action as control target for Stabe PD controller. The simulation will carry on from that point.
##Additionally, we also feed agent several other states for monitoring different things.
##Impact info tells agent the information about the external physical interactions
##Ground info and feet contacts monitor the ground level around player and feet placement
##We also pass in the action from previous frame to help agent smooth out the action generation.
##Reward wise, similar to DReCon, we use multiple sub rewards to

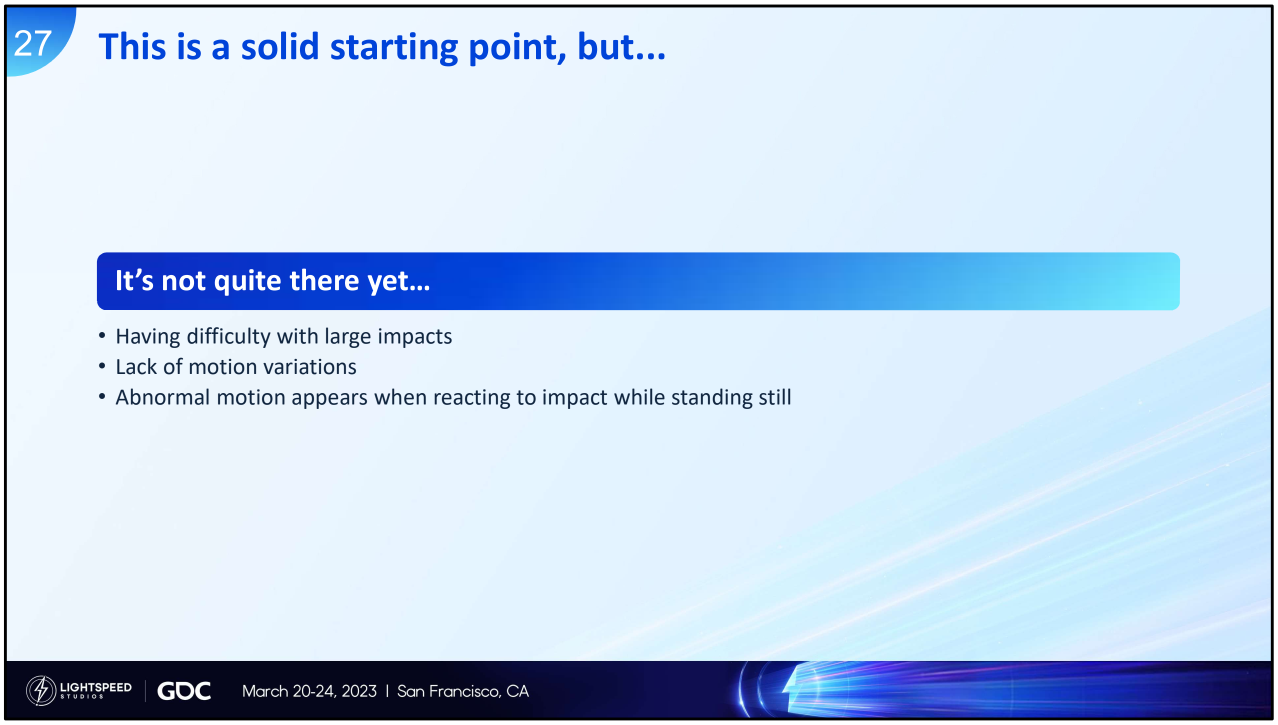
evaluate similarity of root motion, local motion and local pose between ragdoll and target. The sum of all the sub rewards will then be scaled by the falling factor, which is measured by the head vertical distance between the two. And here we go, we get the final reward for our agent.



Early WIP Training Result



So, this demo video shows the early training result when combining DReCon and DeepMimic together. The solid character is agent-controlled ragdoll. As you see here, it can follow the target locomotion, which represented by the transparent character.
It definitely look more promising, however, you might already notice that the impact reaction looks very rigid and repetitive. Its body sometimes twitches weirdly when hit while standing still.


27

This is a solid starting point, but...

It’s not quite there yet...

- Having difficulty with large impacts
- Lack of motion variations
- Abnormal motion appears when reacting to impact while standing still

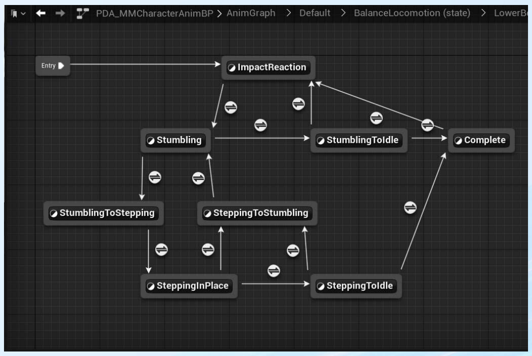
LIGHTSPEED STUDIOS | GDC March 20-24, 2023 | San Francisco, CA

Well, we are not quite there yet. Other than the problem I mentioned earlier, the trained ragdoll also has difficulties to handle large impacts, such as bullet shots.
So how are we going improve this?

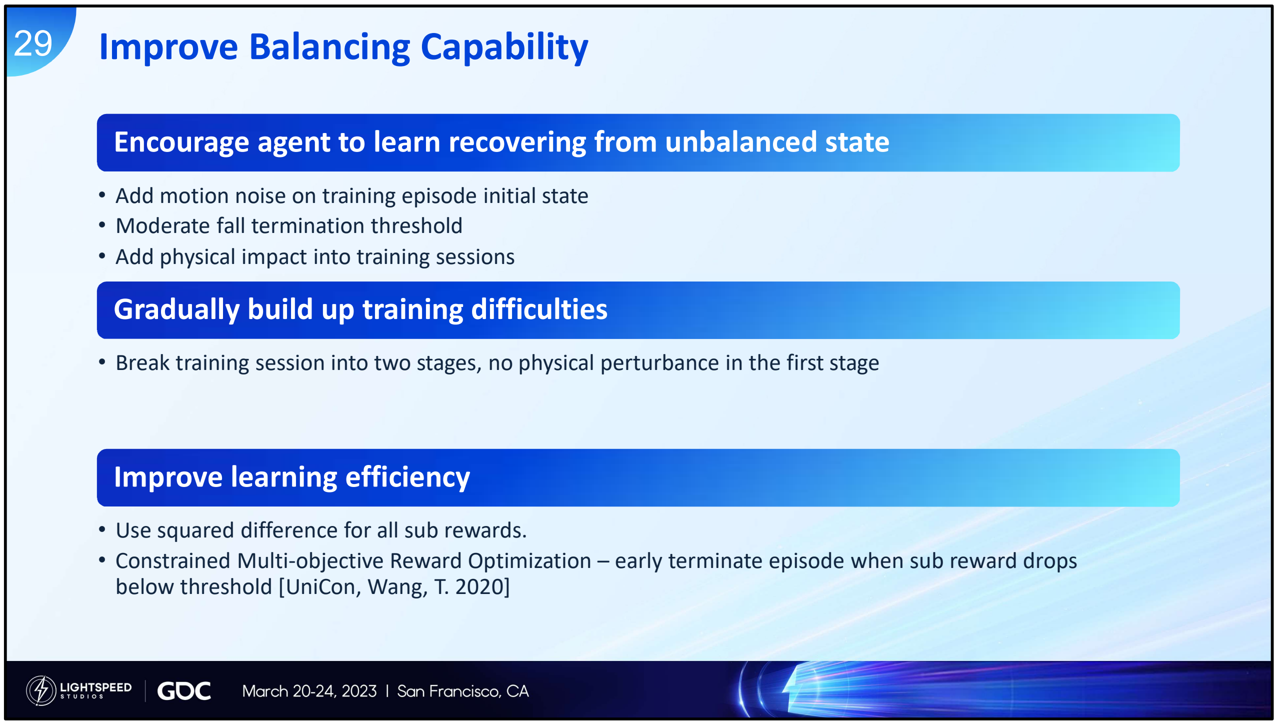
28

Improve Motion Variations

- Enlarge motion compacity
- Created balancing locomotion system
- Added cosmetic reaction behaviors on upper body
- Post-physics animation composition



First thing first, we added balancing motion into the training. We tried both one off motions and looped stumbling motions. It turned out the agent can pick up looped motion more effectively. Therefore, we built a state machine for handling balancing locomotion as shown in the picture on the right. Both stumbling and settling are looping motions. They get connected with several transition motions. Another thing we tried is using the partial body motion and post-physics blend for cosmetic movements. They help adding more variations and liveness to the reaction motion.



29

Improve Balancing Capability

Encourage agent to learn recovering from unbalanced state

- Add motion noise on training episode initial state
- Moderate fall termination threshold
- Add physical impact into training sessions

Gradually build up training difficulties

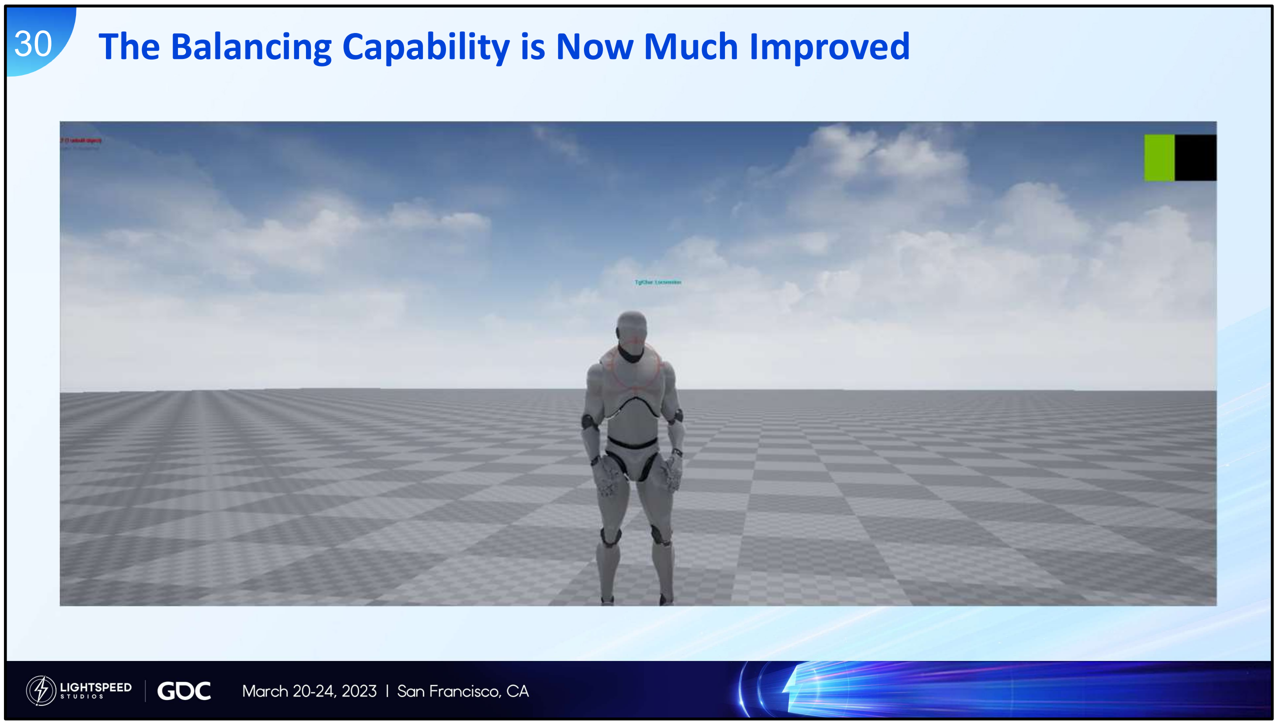
- Break training session into two stages, no physical perturbance in the first stage

Improve learning efficiency

- Use squared difference for all sub rewards.
- Constrained Multi-objective Reward Optimization – early terminate episode when sub reward drops below threshold [UniCon, Wang, T. 2020]

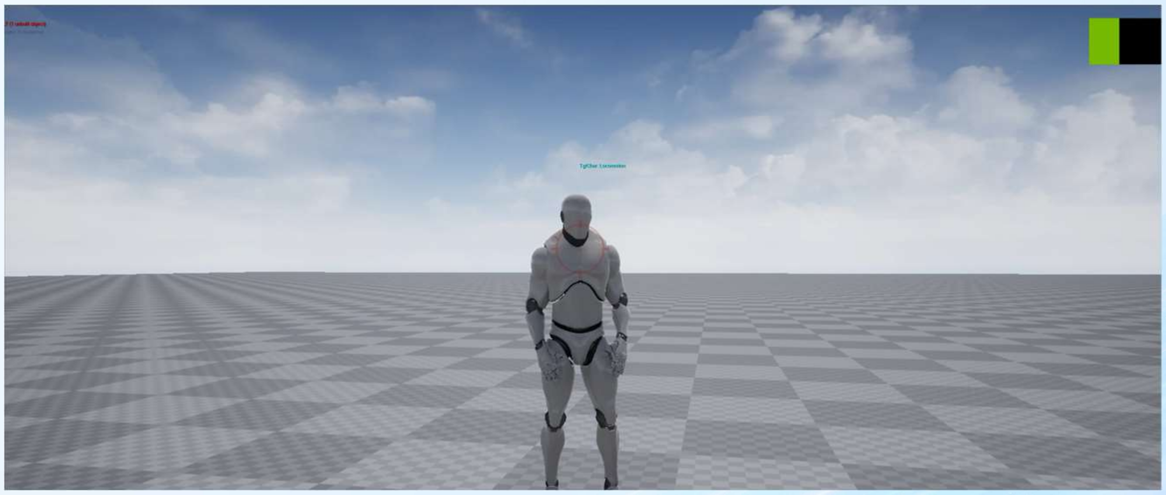
LIGHTSPEED STUDIOS | GDC March 20-24, 2023 I San Francisco, CA

For improving balancing capability, we did three things.
First, we exposed agent to unbalanced state a lot more and encourage it to learn how to recover. We did so by adding more motion noise, physical impact; also defer the fall termination.
Secondly, we staggered the training process with different difficulties. This is because we found that the learning progress became very slow if we add too much motion noise right from start.
Lastly, we improved learning efficiency by playing with reward calculation and used a trick introduced by UniCon, called Constrained Multi-objective Reward Optimization. What it does is early terminates episode when sub reward drops below the threshold. This trick helps balancing the weights between different sub rewards.



30

The Balancing Capability is Now Much Improved

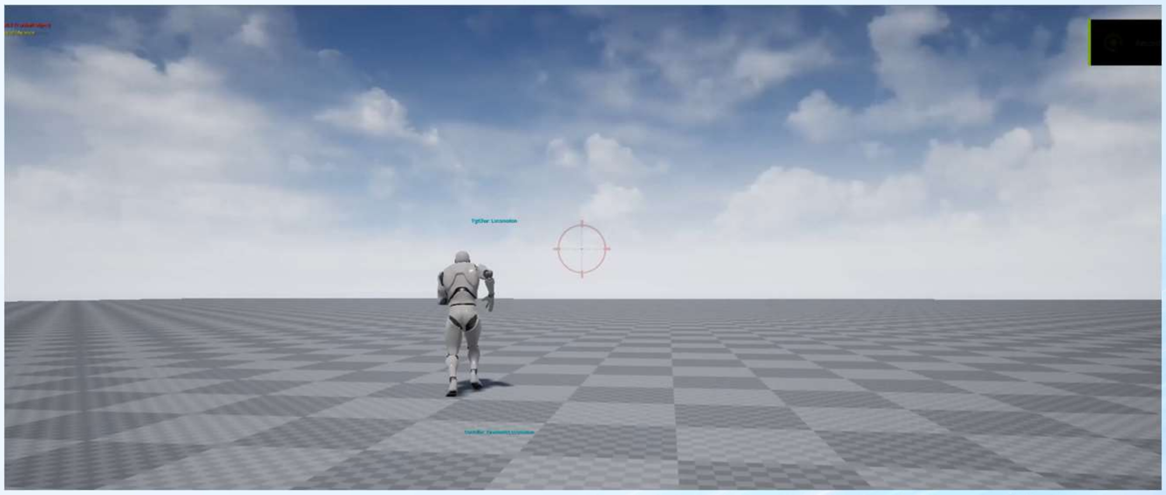


LIGHTSPEED STUDIOS | GDC March 20-24, 2023 | San Francisco, CA

Let’s take a look how does it perform with updated training scheme. Now you have to try really really hard to knock him down. Each bullet impact is 100 newton second (N*s). That’s equivalent to a 9mm bullet(150 grains or 10g) traveling at 10,000m/s. From the impact deformation, you can tell that’s extremely heavy. Even so, it takes more than 20 shots to knock him down.

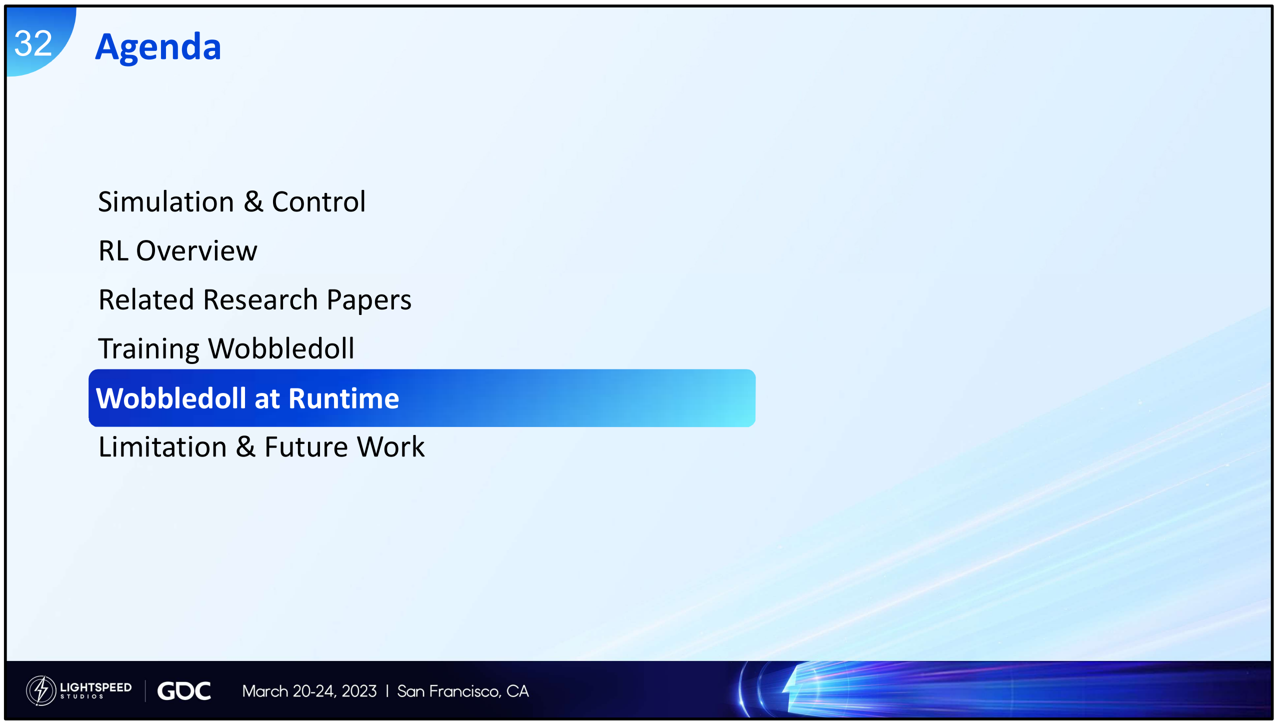
31

Also Supports Variety of Reaction Behaviors



LIGHTSPEED STUDIOS | GDC March 20-24, 2023 | San Francisco, CA

Here's another demo showing the bullet hit reaction while running. Other than amazing balancing capability, the balancing motion is also much more lifelike and polished. Sometimes covers the wound, sometimes swings the arms, sometimes body flinches. Now you really feel the impact each bullet produces.


32

Agenda

Simulation & Control

RL Overview

Related Research Papers

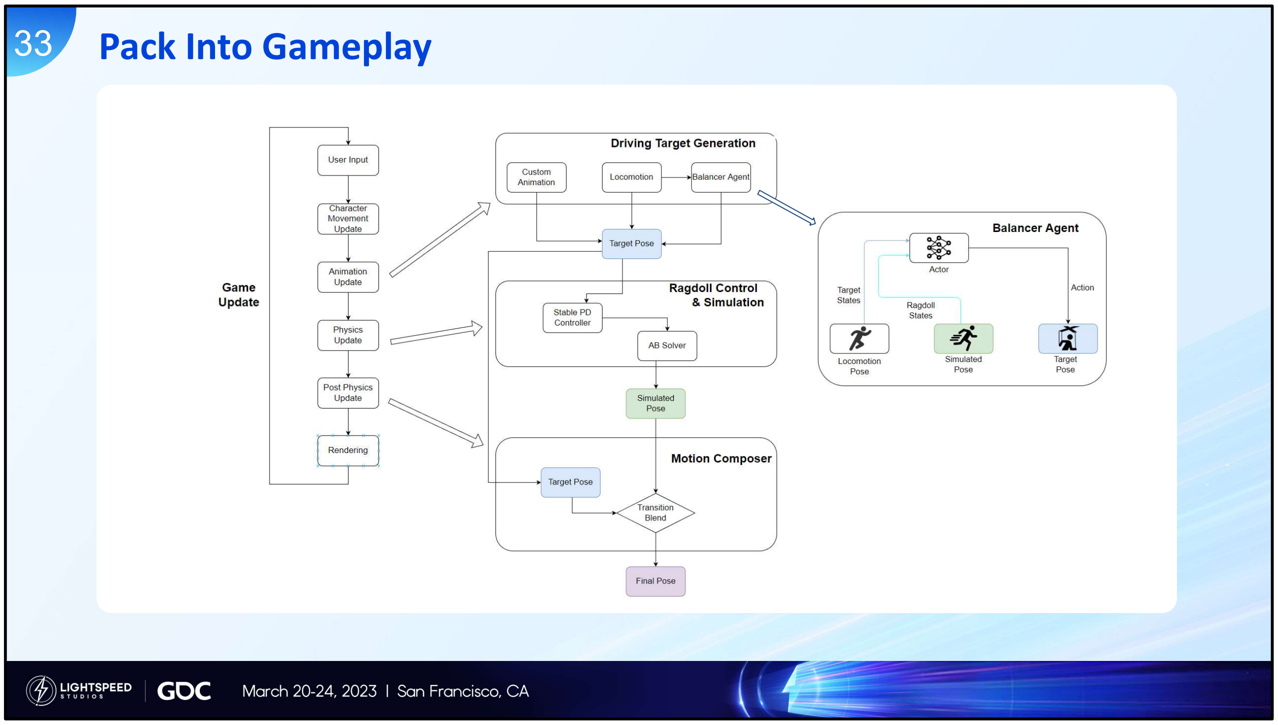
Training Wobbledoll

Wobbledoll at Runtime

Limitation & Future Work

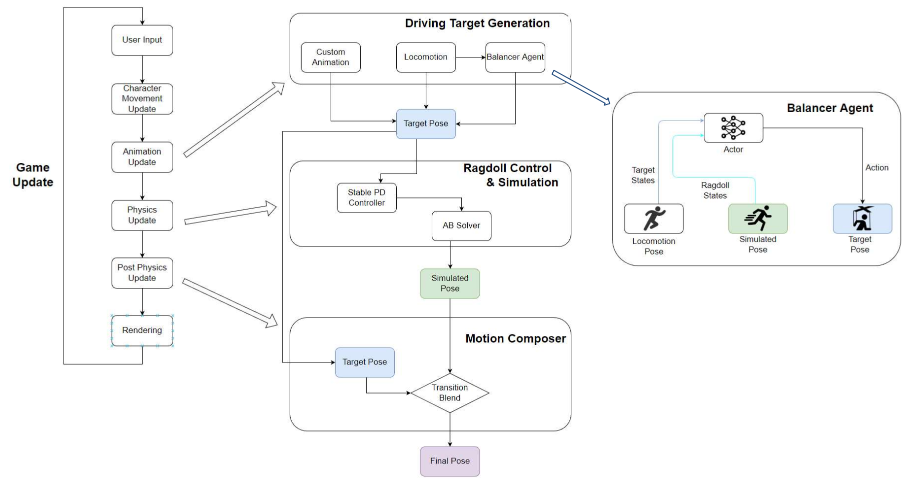
LIGHTSPEED STUDIOS | GDC March 20-24, 2023 I San Francisco, CA

Alright, now we are happy with training result, let’s talk about how to deploy it at runtime.



33

Pack Into Gameplay

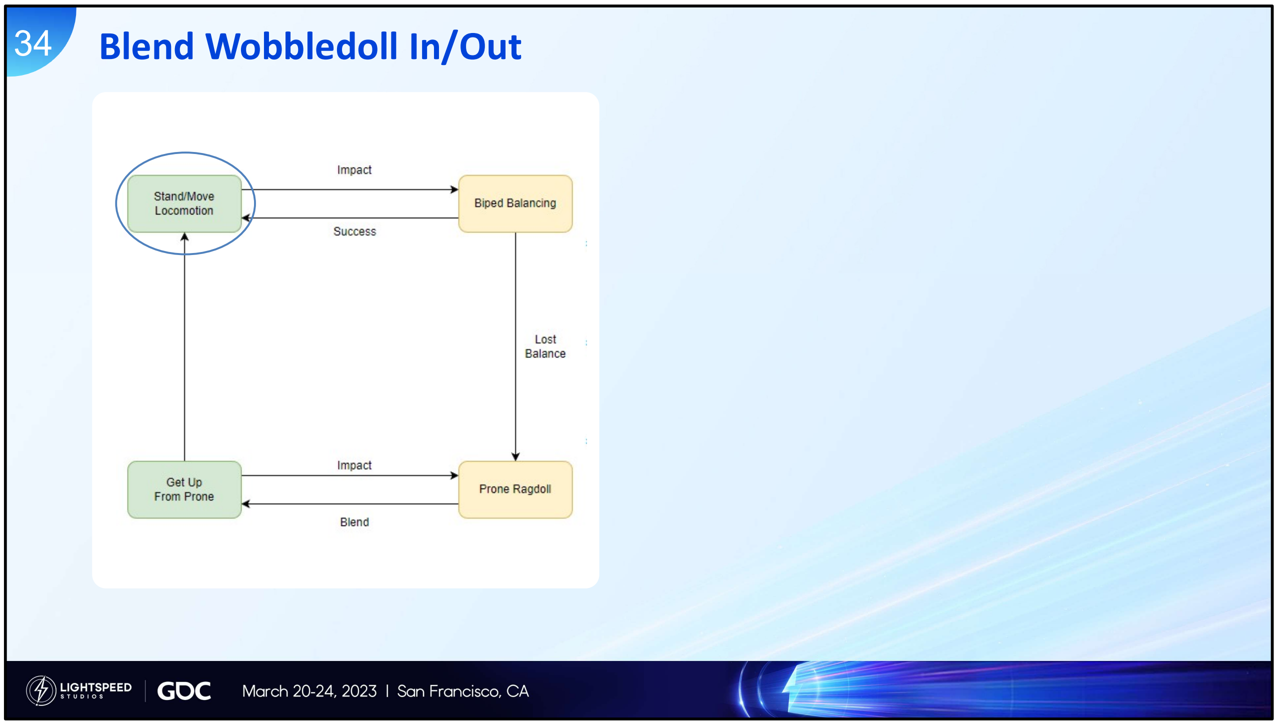


LIGHTSPEED STUDIOS | GDC March 20-24, 2023 I San Francisco, CA

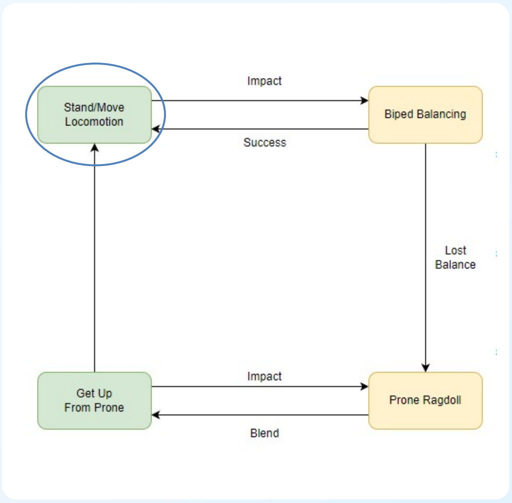
Start with basic game update loop. The physical animation takes place among the animation update, physics update and post physics update. Let’s take a closer look at those ones.

##Now the animation update is our driving target generation phase, which produces the target motion for ragdoll to follow. This is also where the balance agent applies its action to correct the target pose for balancing purposes.
The Ragdoll Control & Simulation phase happens in physics update, where it drive the ragdoll to follow the given target motion.
Finally, in post physics update, we perform any post blend with the updated ragdoll pose and animation pose or IK correction. This phase is useful for blending between physical animation and kinematic animation states.
##Now let’s look further into balancer agent for a second. When

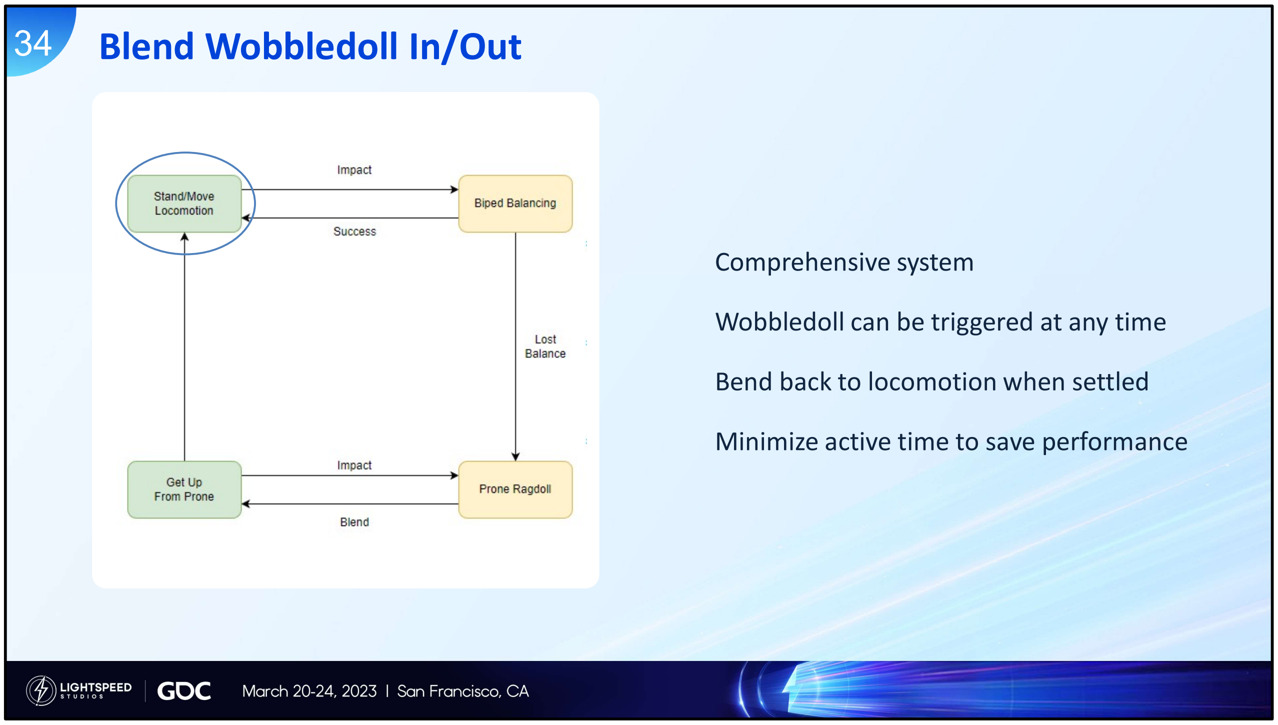
agent is fully trained, all we need is its policy network, which can be used as a local function module. This is so called local inference. Afterall, this whole chart basically covers the high-level overview of the Wobbledoll system in-game integration. Next, I will talk about how it can work cohesively with kinematic animation system.



Blend Wobbledoll In/Out

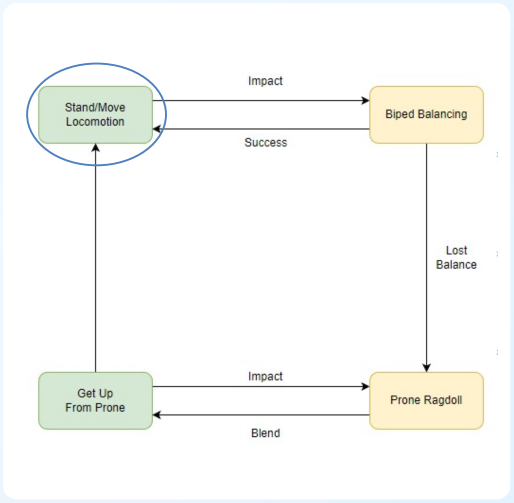


The idea here is that we want to keep the game character stay in the kinematic state as much as possible to keep tight control and save performance cost. The Wobbledoll only takes over when there's a physical interaction happened. To do so, we built mechanics to transition between two systems seamlessly.
Here's a quick example of how the logic flow works.
##Let's say a character starts with playing locomotion, moving around and doing whatever.
##Then it got hit by something, the Wobbledoll system will be activated to react and try to regain the balance. If it successfully do so, it would blend back to locomotion and that's it, end of show.
##But if the character got knocked over, the balancer will be shut down and only play the physical animation.



34

Blend Wobbledoll In/Out



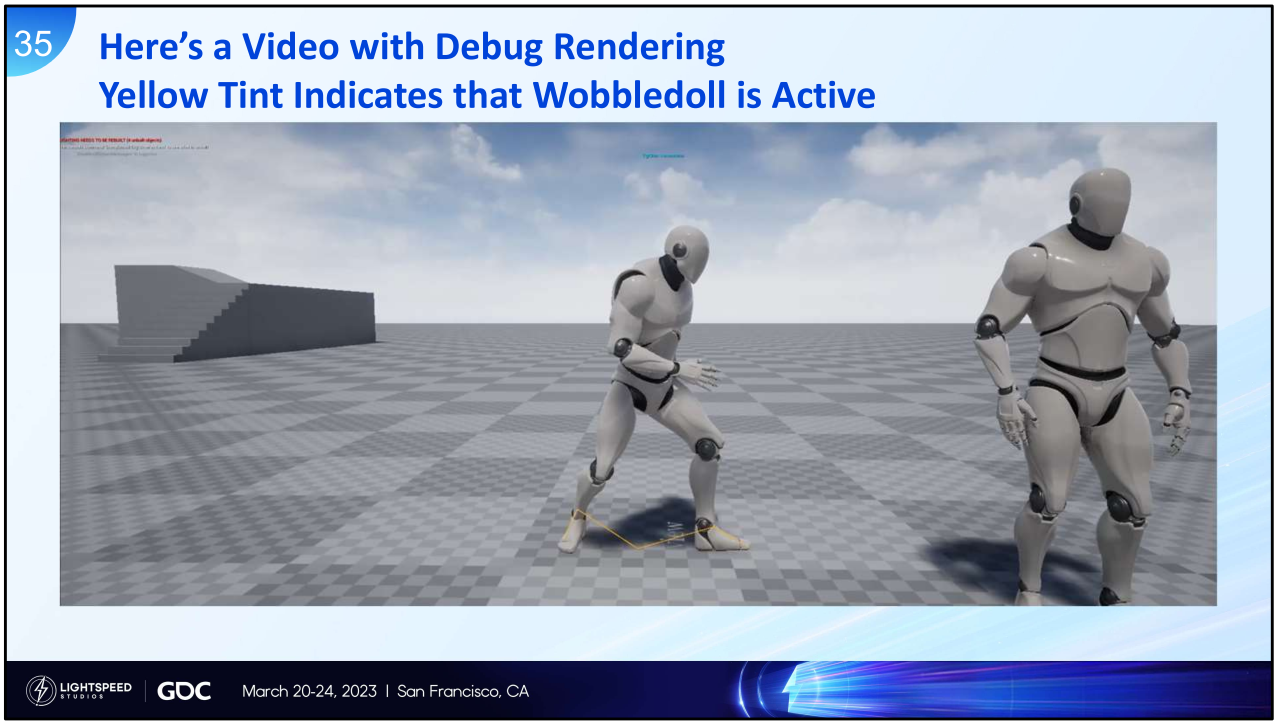
Comprehensive system

Wobbledoll can be triggered at any time

Bend back to locomotion when settled

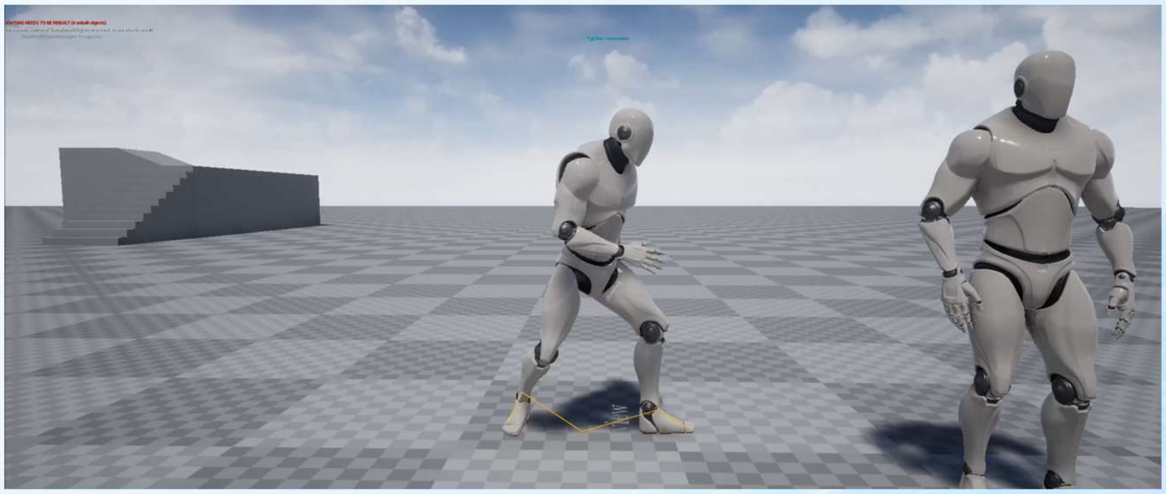
Minimize active time to save performance

##Once it settle on the ground. We can then blend it black to kinematic get-up animation.
There could be cases where the character got hit again while getting up. Wobbledoll might enable balancer if character is already getting off the ground, otherwise, we would just let it fall again.
##The cool thing about this logic flow is that we basically turning the Wobbledoll into a comprehensive system, which means it covers pretty much all possible scenarios, where character physical interaction could happen. And it is performance efficient as the Wobbledoll only stays active for a brief moment, and then it would blend back to kinematic state when motion settles.



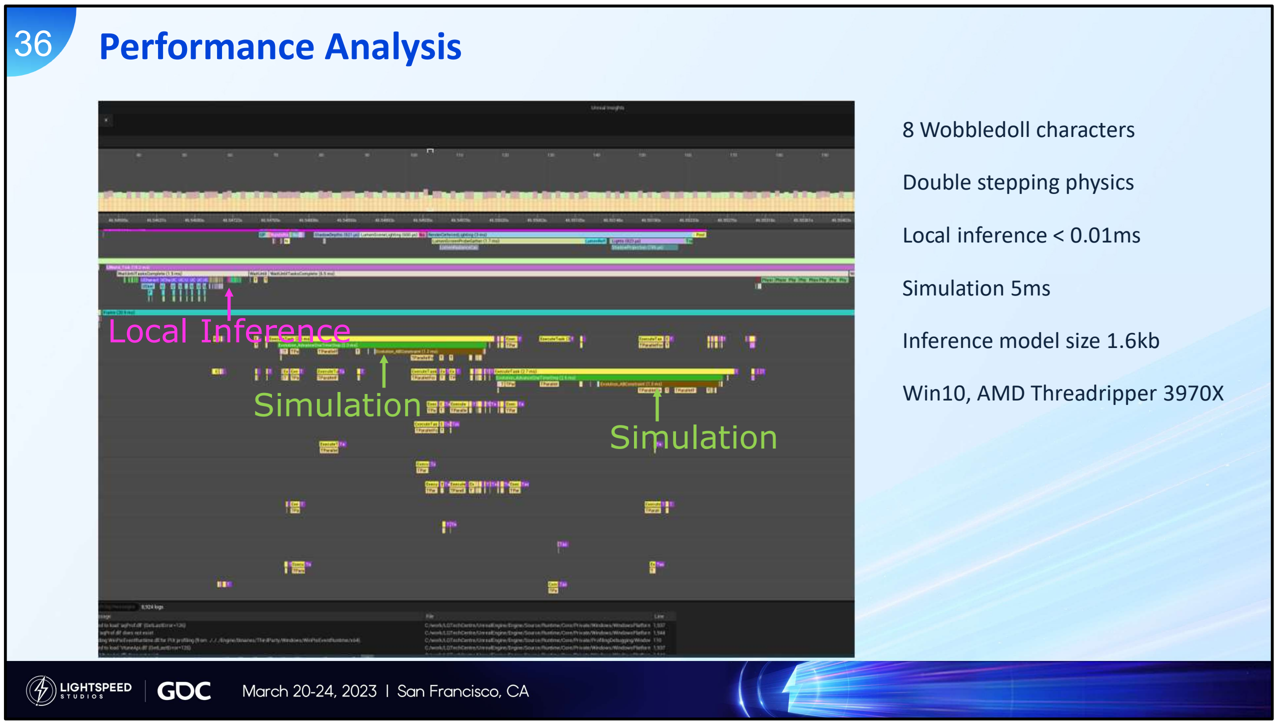
35

Here's a Video with Debug Rendering
Yellow Tint Indicates that Wobbledoll is Active



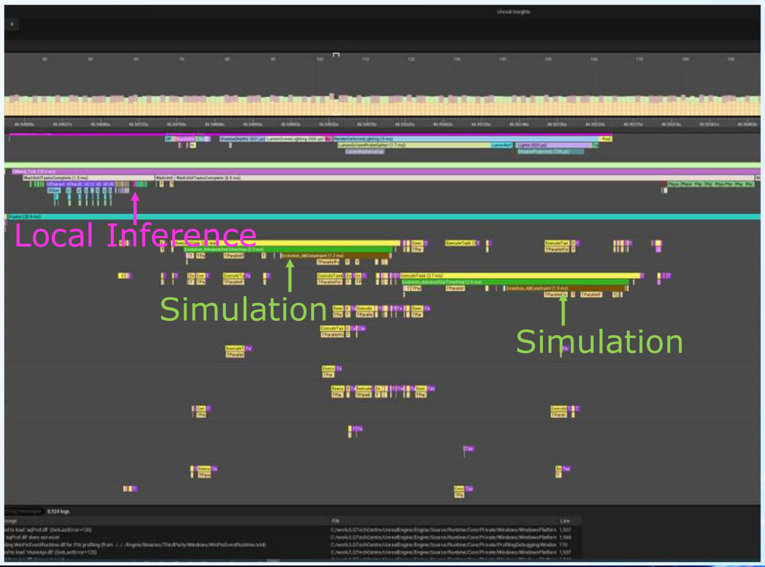
LIGHTSPEED STUDIOS | GDC March 20-24, 2023 | San Francisco, CA

Let's take a look at this video with debug rendering on. The character would turn yellow when the Wobbledoll is alive. As you can see, the NPC activates Wobbledoll once player is bumping into him. Then it blend back to kinematic when motion settles. The transition between the two is pretty much seamless.



36

Performance Analysis



8 Wobbledoll characters

Double stepping physics

Local inference < 0.01ms

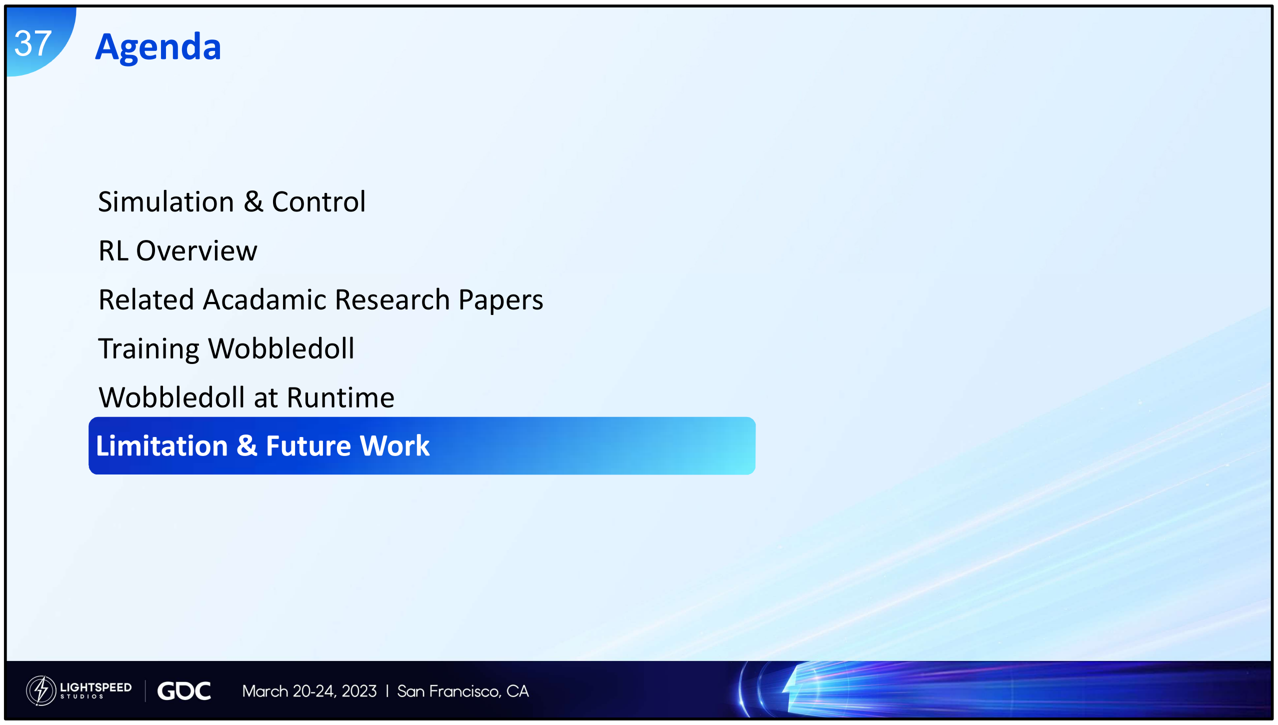
Simulation 5ms

Inference model size 1.6kb

Win10, AMD Threadripper 3970X

LIGHTSPEED STUDIOS | GDC March 20-24, 2023 | San Francisco, CA

what about the runtime cost of Wobbledoll itself? Let's take a look at this profile capture here. It turned out the performance cost is pretty much the same as running a plain ragdoll. That little tiny pink block the arrow pointed at is the time cost for local inference, which is basically nothing compares to the ragdoll simulation. So the wobbleddoll performance is basically bounded by physics simulation. As long as we don't turn on too many of them at the same time, it should be fine.
The inference model size is 1.6KB, which is fairly light weight

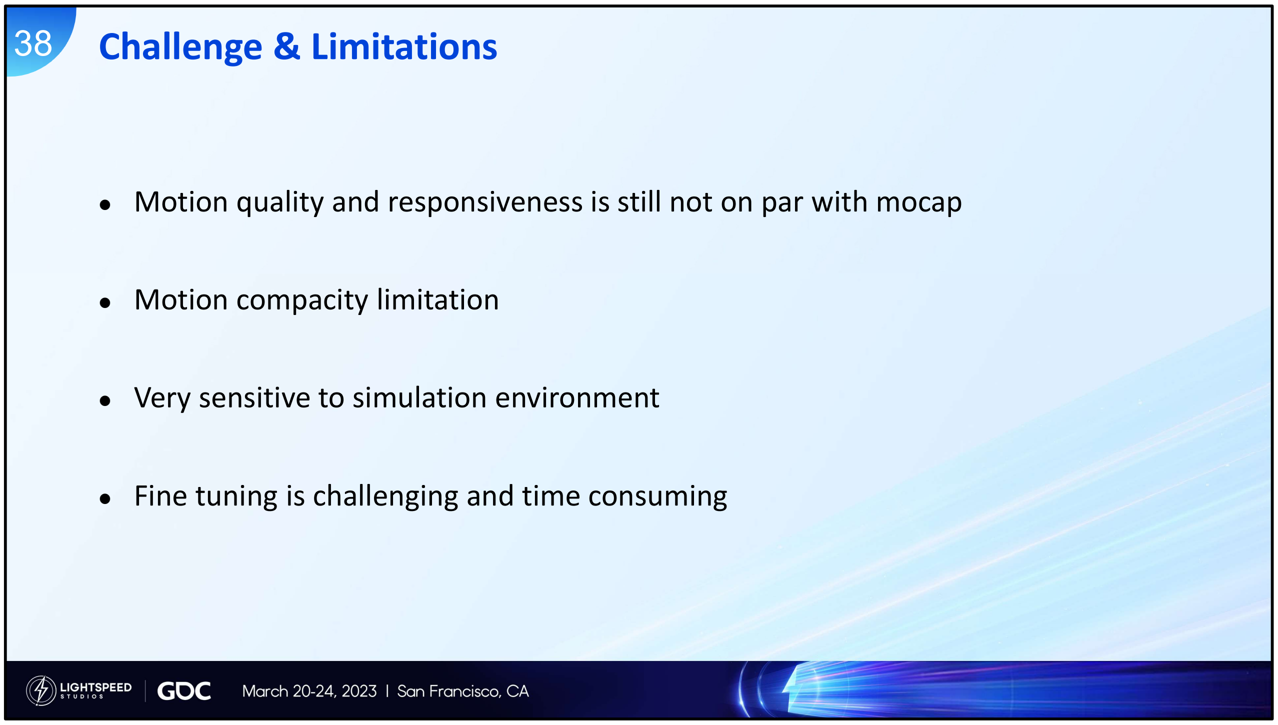

37

Agenda

Simulation & Control
RL Overview
Related Acadamic Research Papers
Training Wobbledoll
Wobbledoll at Runtime
Limitation & Future Work

LIGHTSPEED STUDIOS | GDC March 20-24, 2023 | San Francisco, CA

Lastly, let’s talk about the limitation and future work.



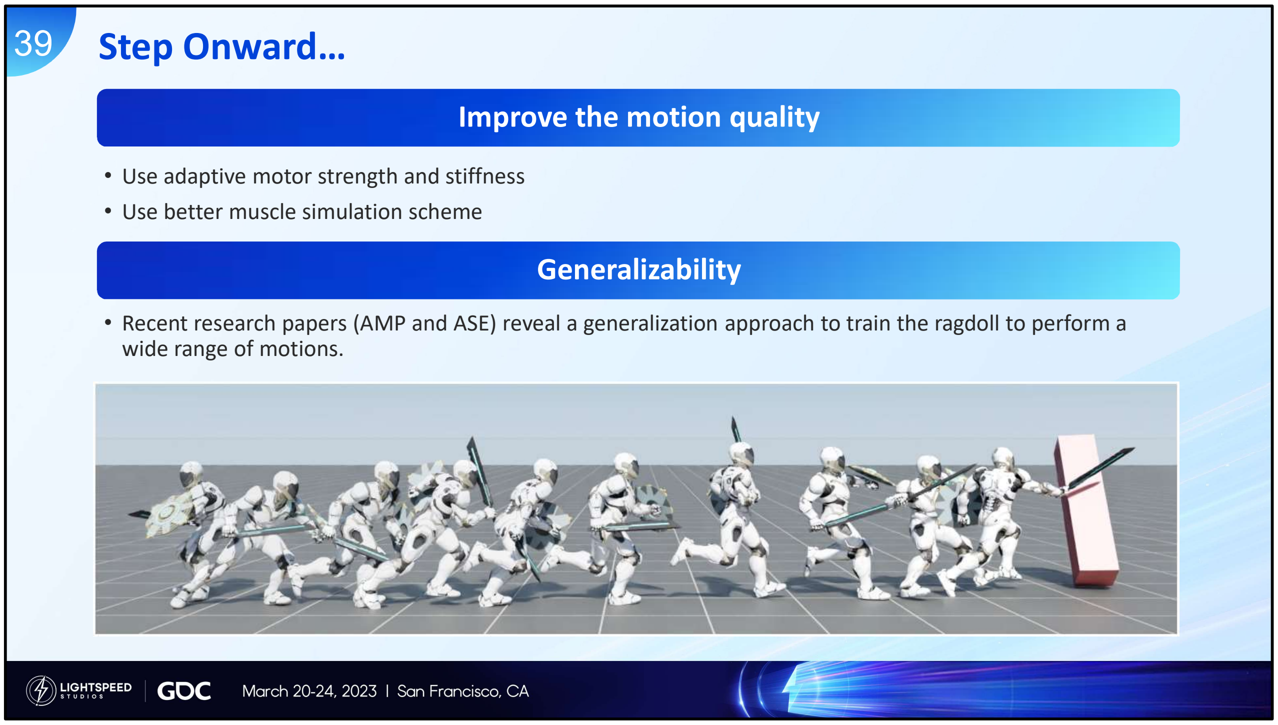
38

Challenge & Limitations

- Motion quality and responsiveness is still not on par with mocap
- Motion compacity limitation
- Very sensitive to simulation environment
- Fine tuning is challenging and time consuming

LIGHTSPEED STUDIOS | GDC March 20-24, 2023 I San Francisco, CA

The Wobbledoll system of course is far from being perfect. It has its own limitations and there are still a lot rooms for improvements. The motion quality is still not ideal. The motion compacity is limited. The system is sensitive to simulation environment and hard to be fine tuned. Sometimes it’s hard to tell if tuning a specific parameter make it better or worth.



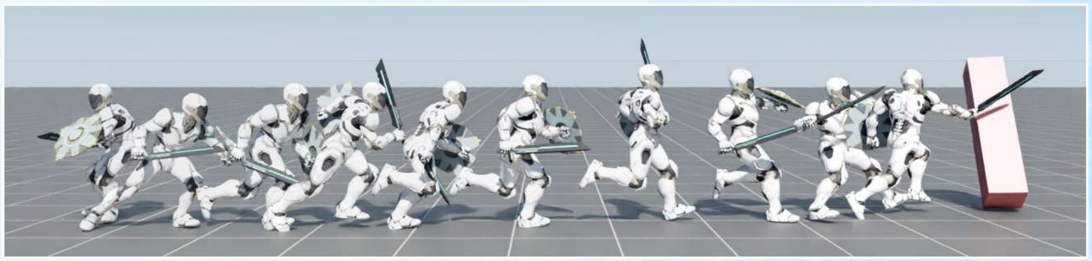
Step Onward…

Improve the motion quality

- Use adaptive motor strength and stiffness
- Use better muscle simulation scheme

Generalizability

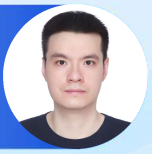
- Recent research papers (AMP and ASE) reveal a generalization approach to train the ragdoll to perform a wide range of motions.



Step onward we are planning to improve Wobbledoll system in two main aspects.
One of them is to improve the motion quality. One thing we are currently experimenting is to see if it is possible to let agent to control the motor strength and stiffness.
Another direction we are looking into is see how to improve the generalizability. A few recent papers reveal a new approach to use generative adversarial network. We are going to experiment with those ideas as well

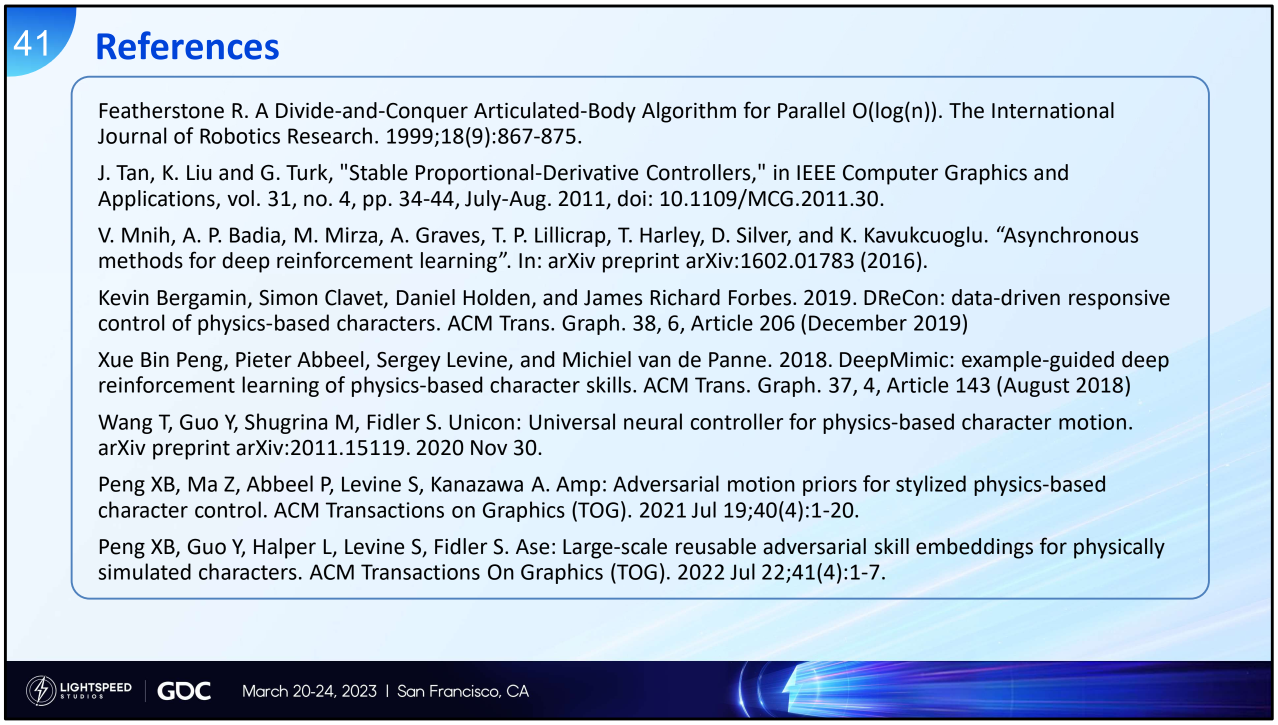
Special Thanks to

Kan Xu
Animation Specialist

Zherong Pan
Robotics/Simulation Specialist


Ryan Zhang
AI Specialist

Lastly, I'd like to take a moment and thanks my teammates, Kan Xu, Zherong Pan and Ryan Zhang. They made great contribution to this research topic. Special thanks to them.



41

References

Featherstone R. A Divide-and-Conquer Articulated-Body Algorithm for Parallel O(log(n)). The International Journal of Robotics Research. 1999;18(9):867-875.

J. Tan, K. Liu and G. Turk, "Stable Proportional-Derivative Controllers," in IEEE Computer Graphics and Applications, vol. 31, no. 4, pp. 34-44, July-Aug. 2011, doi: 10.1109/MCG.2011.30.

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. “Asynchronous methods for deep reinforcement learning”. In: arXiv preprint arXiv:1602.01783 (2016).

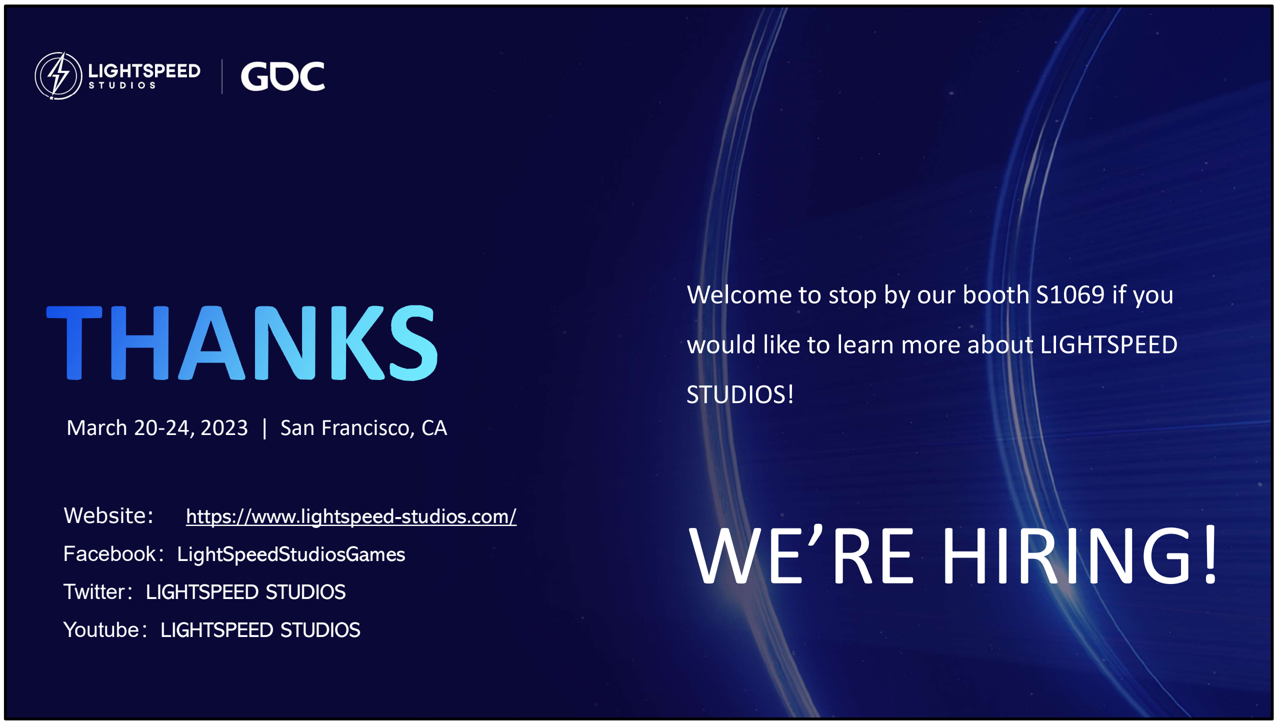
Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. 2019. DReCon: data-driven responsive control of physics-based characters. ACM Trans. Graph. 38, 6, Article 206 (December 2019)

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018. DeepMimic: example-guided deep reinforcement learning of physics-based character skills. ACM Trans. Graph. 37, 4, Article 143 (August 2018)

Wang T, Guo Y, Shugrina M, Fidler S. Unicon: Universal neural controller for physics-based character motion. arXiv preprint arXiv:2011.15119. 2020 Nov 30.

Peng XB, Ma Z, Abbeel P, Levine S, Kanazawa A. Amp: Adversarial motion priors for stylized physics-based character control. ACM Transactions on Graphics (TOG). 2021 Jul 19;40(4):1-20.

Peng XB, Guo Y, Halper L, Levine S, Fidler S. Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters. ACM Transactions On Graphics (TOG). 2022 Jul 22;41(4):1-7.



LIGHTSPEED STUDIOS | GDC

THANKS

March 20-24, 2023 | San Francisco, CA

Website: https://www.lightspeed-studios.com/

Facebook: LightSpeedStudiosGames

Twitter: LIGHTSPEED STUDIOS

Youtube: LIGHTSPEED STUDIOS

Welcome to stop by our booth S1069 if you would like to learn more about LIGHTSPEED STUDIOS!

WE’RE HIRING!