SUPERCHARGED!

Vehicle Physics in Skylanders

Patrick Donnelly & Jan-Erik Steel
Senior Software Engineers
Vicarious Visions

SKYLANDERS


SKYLANDERS

SKYLANDERS

OVERVIEW

OVERVIEW

- *Prototypes*

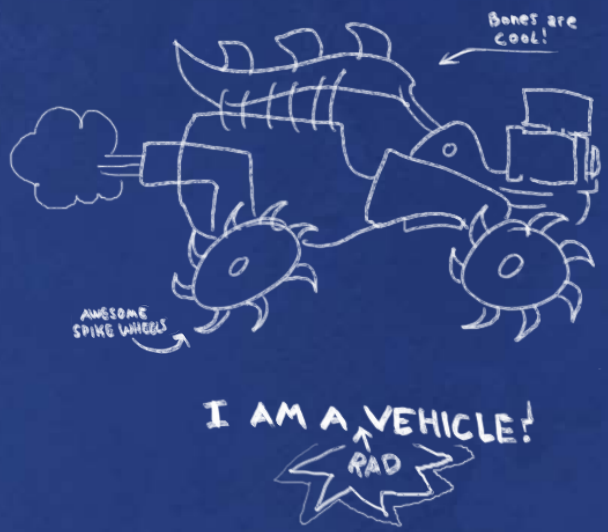
OVERVIEW

- *Prototypes*
- *Our Pillars*
  - *Examples*

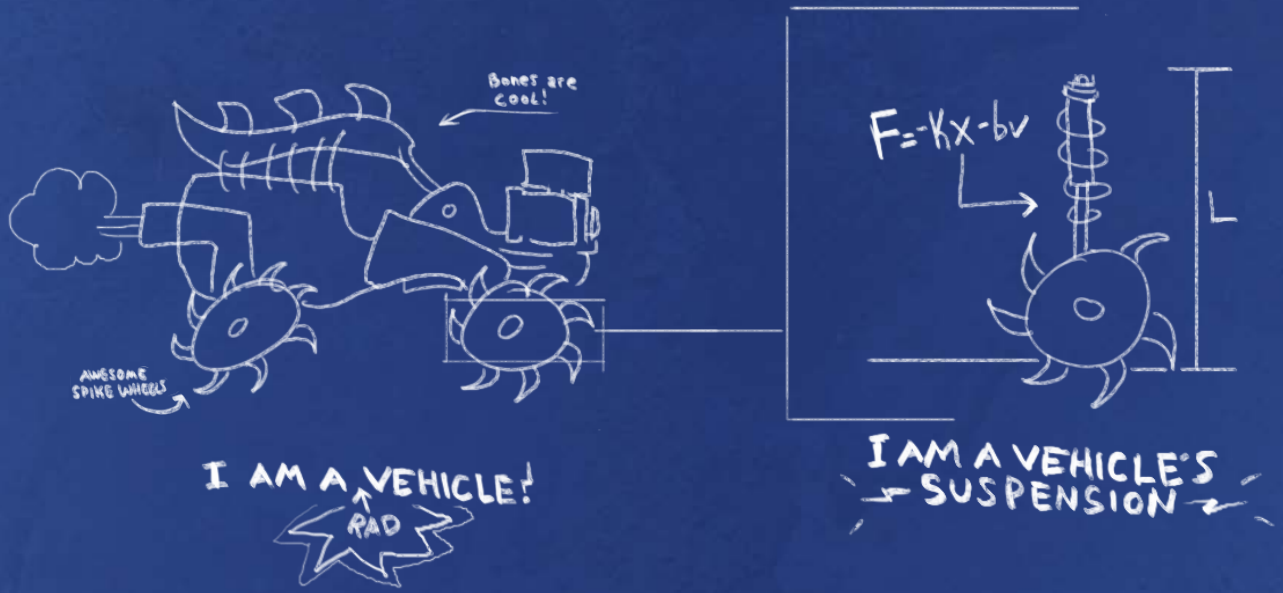
OVERVIEW

- *Prototypes*
- *Our Pillars*
  - *Examples*
- *Technical Deep Dive*

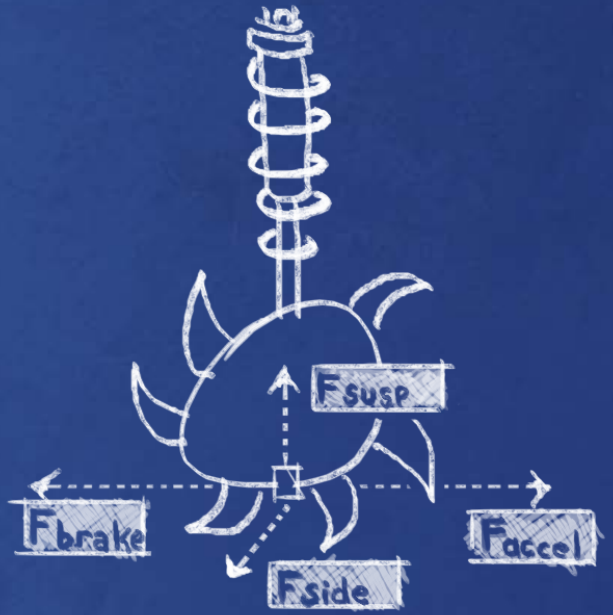
PHYSICS VEHICLES 1.0



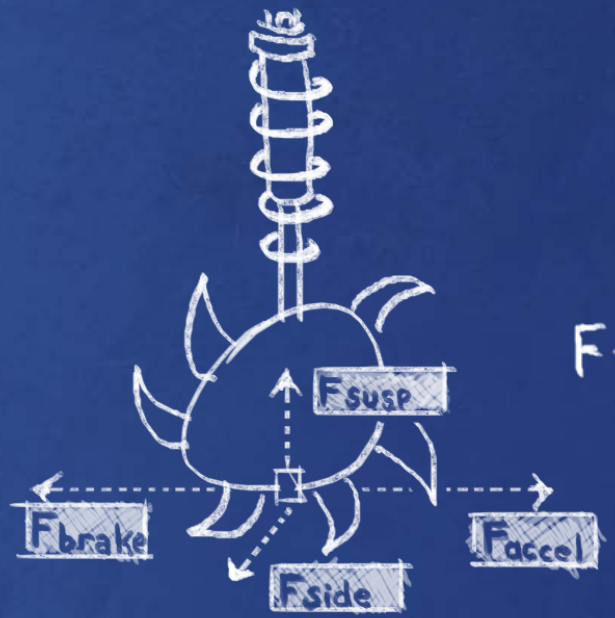
PHYSICS VEHICLES 1.0



PHYSICS VEHICLES 1.0



PHYSICS VEHICLES 1.0

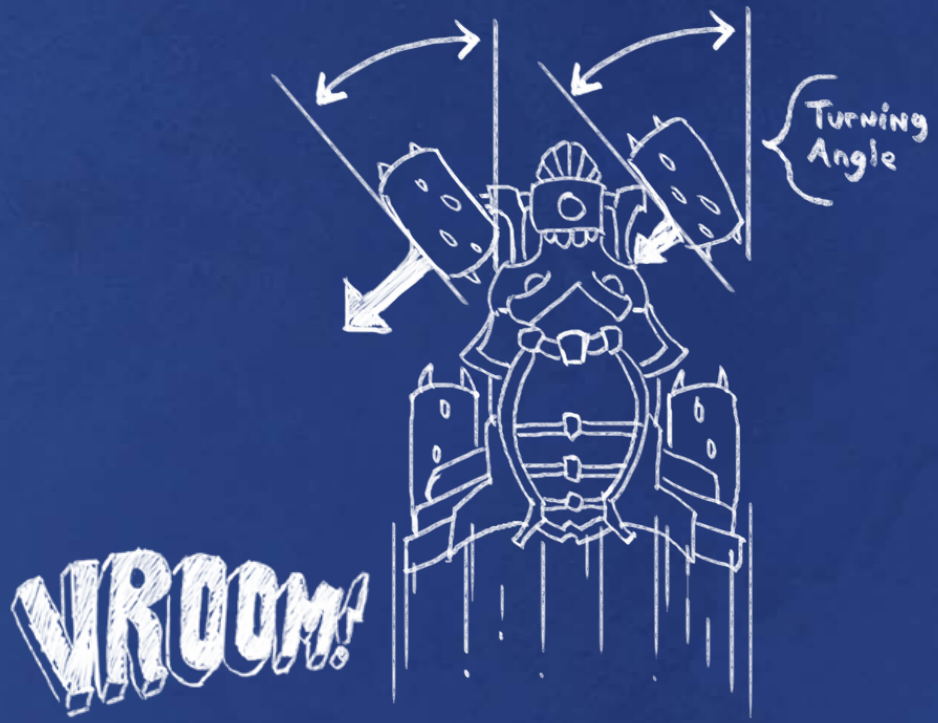


$$|F_{GRIP}| = |F_{SUSP}| \cdot K_{GRIP}$$

$$F_{WHEEL} = F_{ACCEL} + F_{BRAKE} + F_{SIDE}$$

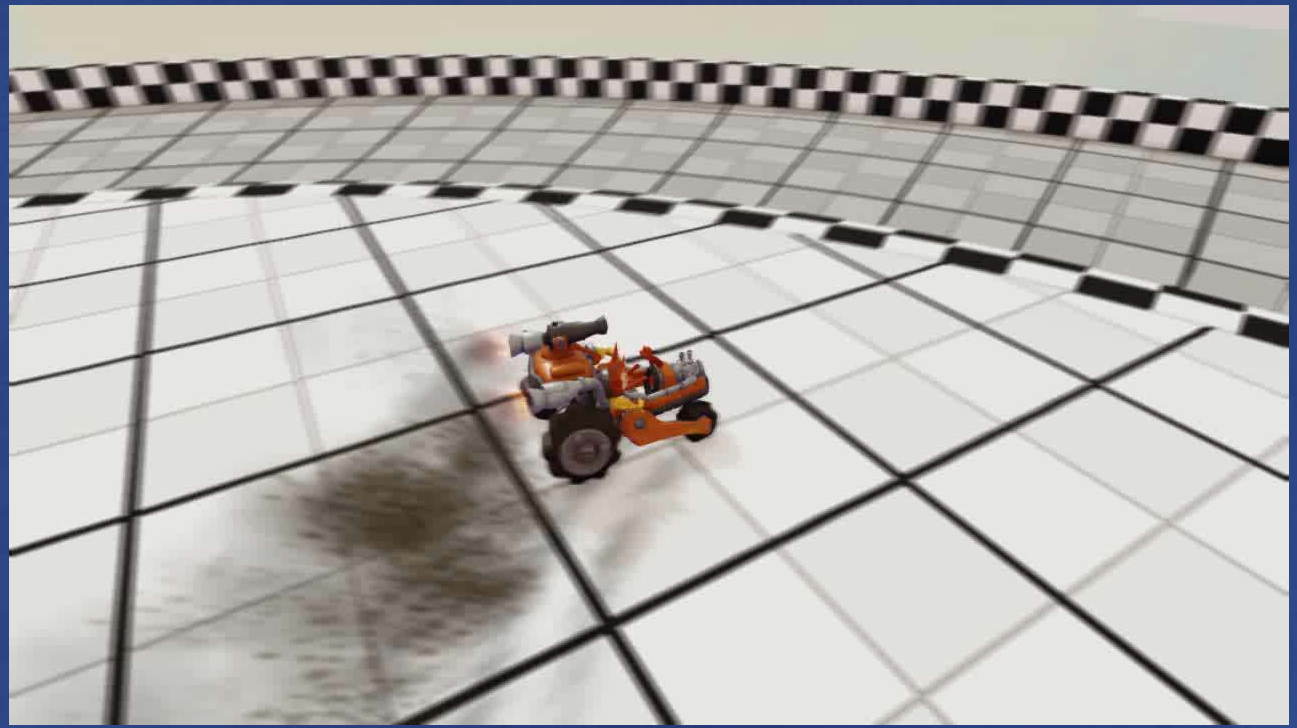
$$F_{TOTAL} = F_{WHEEL} \cdot \frac{\min(|F_{GRIP}|, |F_{WHEEL}|)}{|F_{WHEEL}|} + F_{SUSP}$$

PHYSICS VEHICLES 1.0



PHYSICS VEHICLES 1.0

PHYSICS VEHICLES 1.0



A DESIGN NIGHTMARE

A DESIGN NIGHTMARE

- *Force/acceleration are essentially magic numbers*

A DESIGN NIGHTMARE

- *Force/acceleration are essentially magic numbers*
- *Easy to make vehicles unstable*

A DESIGN NIGHTMARE

- *Force/acceleration are essentially magic numbers*
- *Easy to make vehicles unstable*
- *Difficult to make vehicle feel unique*

CHARACTER VEHICLES

CHARACTER VEHICLES

- *Leverage our robust character pipeline*

CHARACTER VEHICLES

- *Leverage our robust character pipeline*
- *Move away from physically accurate simulation*
- *State machine driven motion*
- *Animation layering to recapture physical motion*
- *Simpler designer interface*

MISSED THE MARK

WHAT WE LEARNED

Physics Vehicle

inertia + chaos = FUN!

Handles organic terrain well

Complex motion by layering forces

Character Vehicle

More control over simulation

Designer Friendly ☺

Animation layering = POWERFUL!

Physics Vehicles 2.0:

- *Create a system that has best of both worlds!*

OUR PILLARS

“Everything should be made as simple as possible, but not simpler.”
– Albert Einstein

OUR PILLARS

“Everything should be made as simple as possible, but not simpler.”
– Albert Einstein

Simplify physics simulation when appropriate

OUR PILLARS

“Everything should be made as simple as possible, but not simpler.”
– Albert Einstein

Simplify physics simulation when appropriate

Complex behavior should come from layering simple systems

OUR PILLARS

“Everything should be made as simple as possible, but not simpler.”
– Albert Einstein

Simplify physics simulation when appropriate

Parameters should be translated into designer language

Complex behavior should come from layering simple systems

OUR PILLARS

“Everything should be made as simple as possible, but not simpler.”
– Albert Einstein

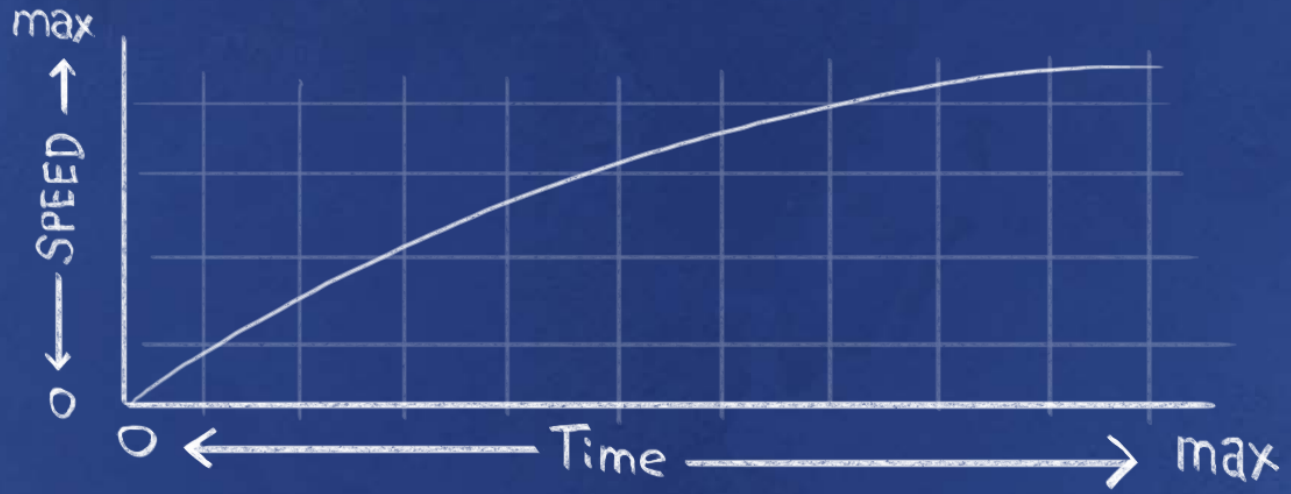
Simplify physics simulation when appropriate

Parameters should be translated into designer language

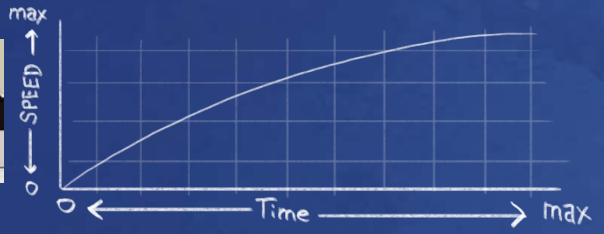
Complex behavior should come from layering simple systems

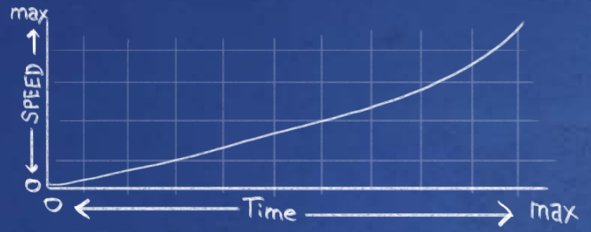
Keep parameters independent

ACCELERATING TO TOP SPEED

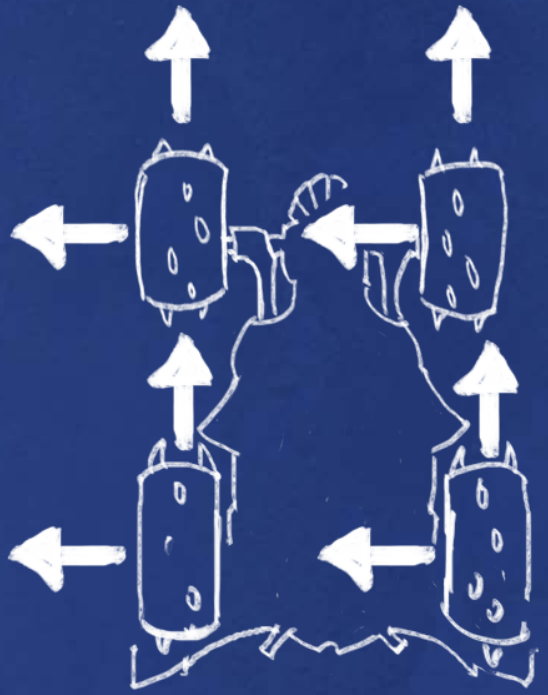


ACCELERATING TO TOP SPEED



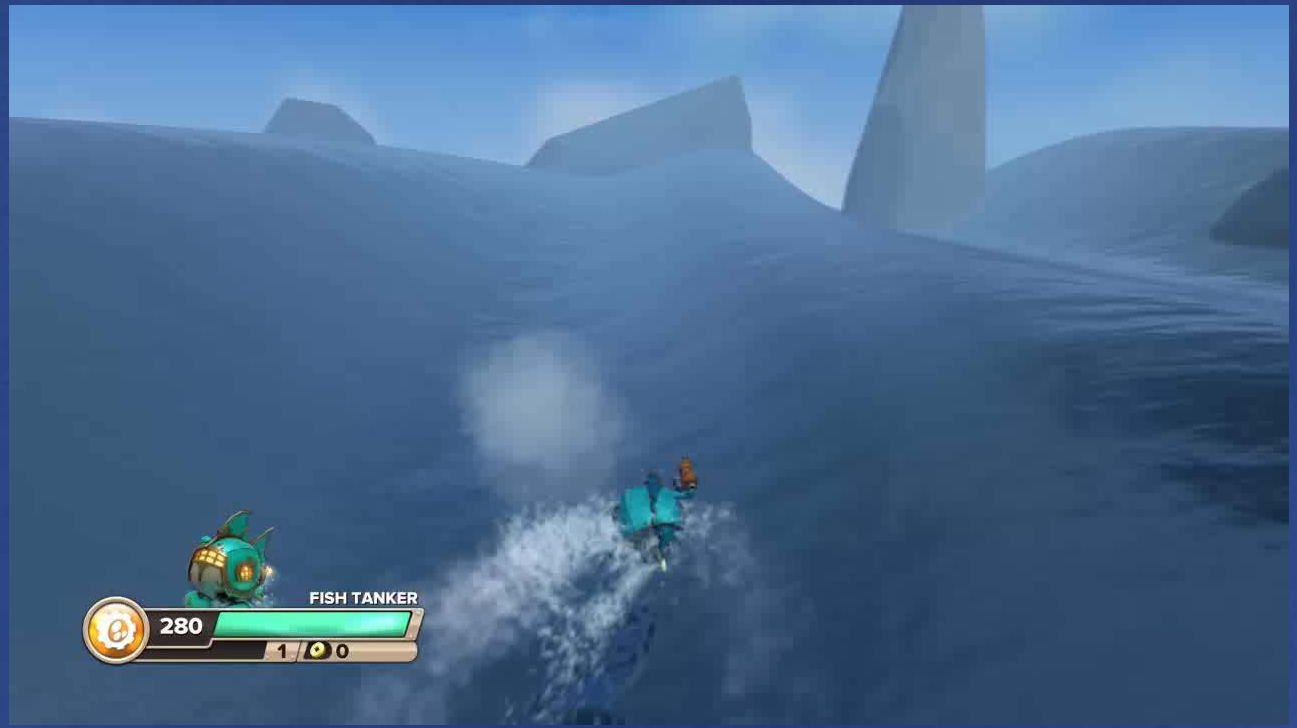


REDUCING COMPLEXITY



REDUCING COMPLEXITY

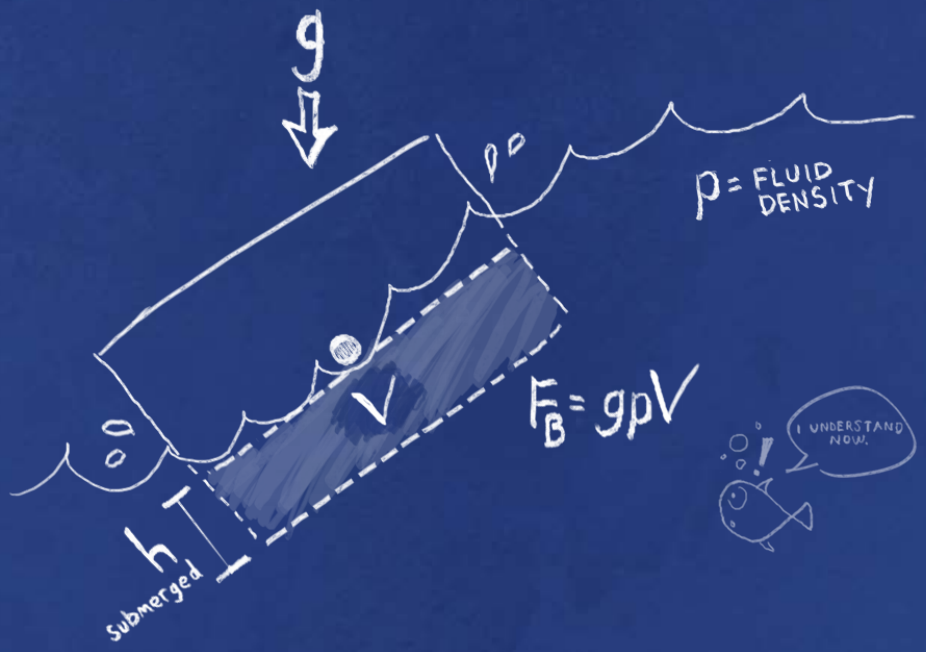
BUOYANCY



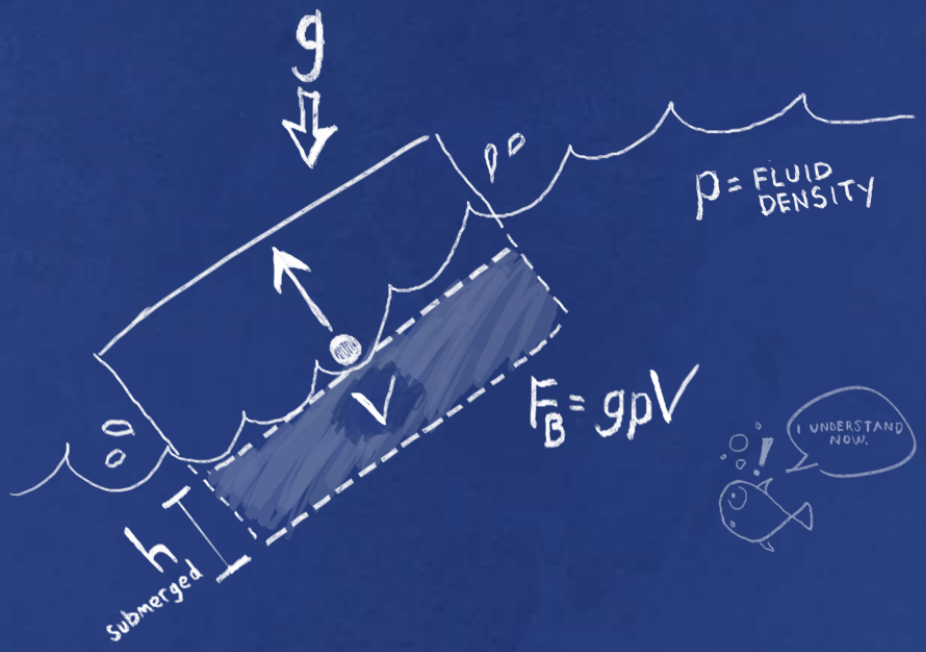
MODELING BUOYANCY

Buoyancy: A fluid exerts a 'buoyant' force on an object wholly or partially submerged, and the magnitude of that force is equal to the weight of the fluid that is displaced

SIMPLIFIED BUOYANCY MODEL

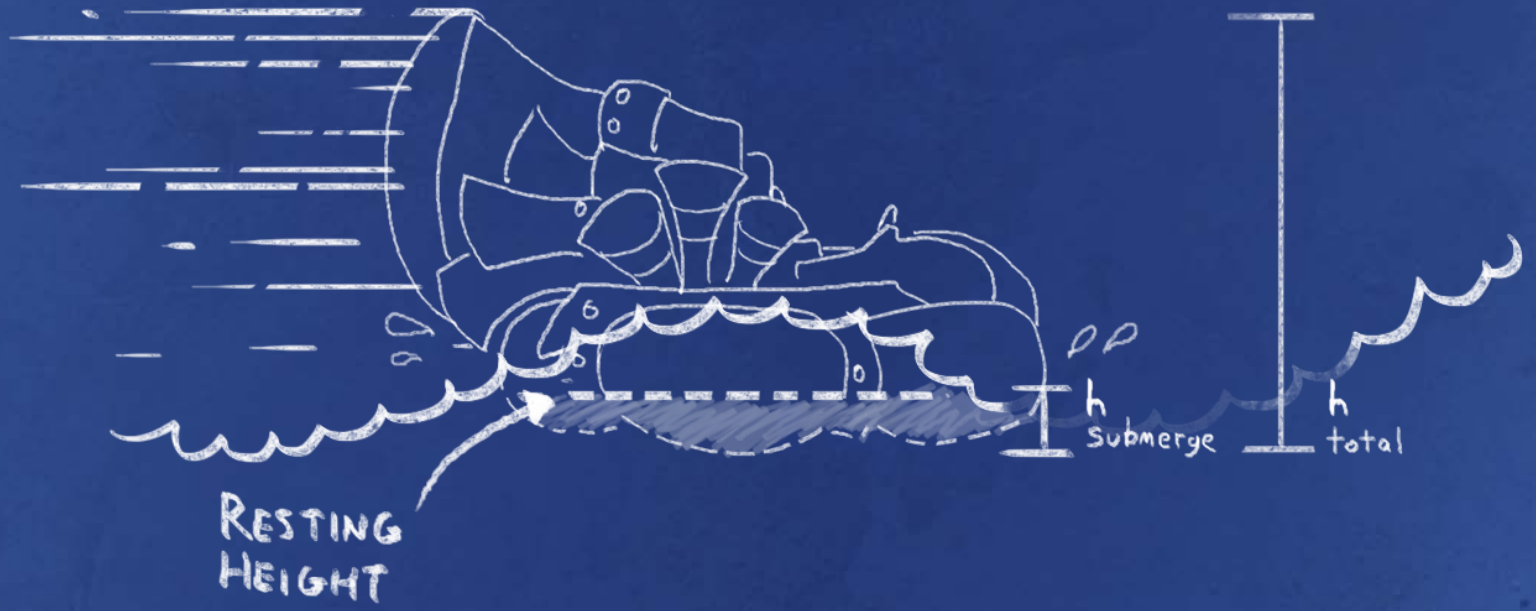


SIMPLIFIED BUOYANCY MODEL



BUOYANCY

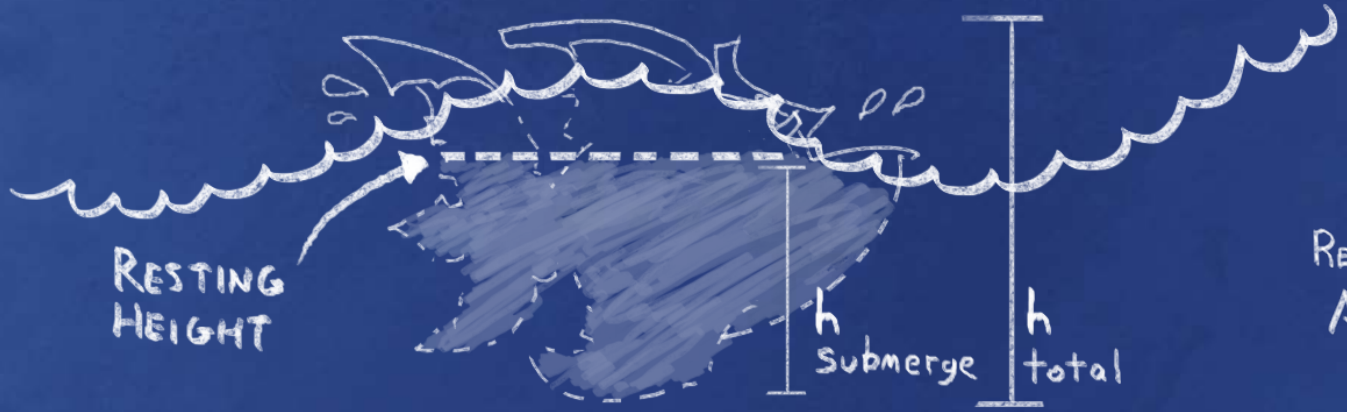
BUOYANCY RESTING HEIGHT



BUOYANCY RESTING HEIGHT

TRANSLATE

INDEPENDENT



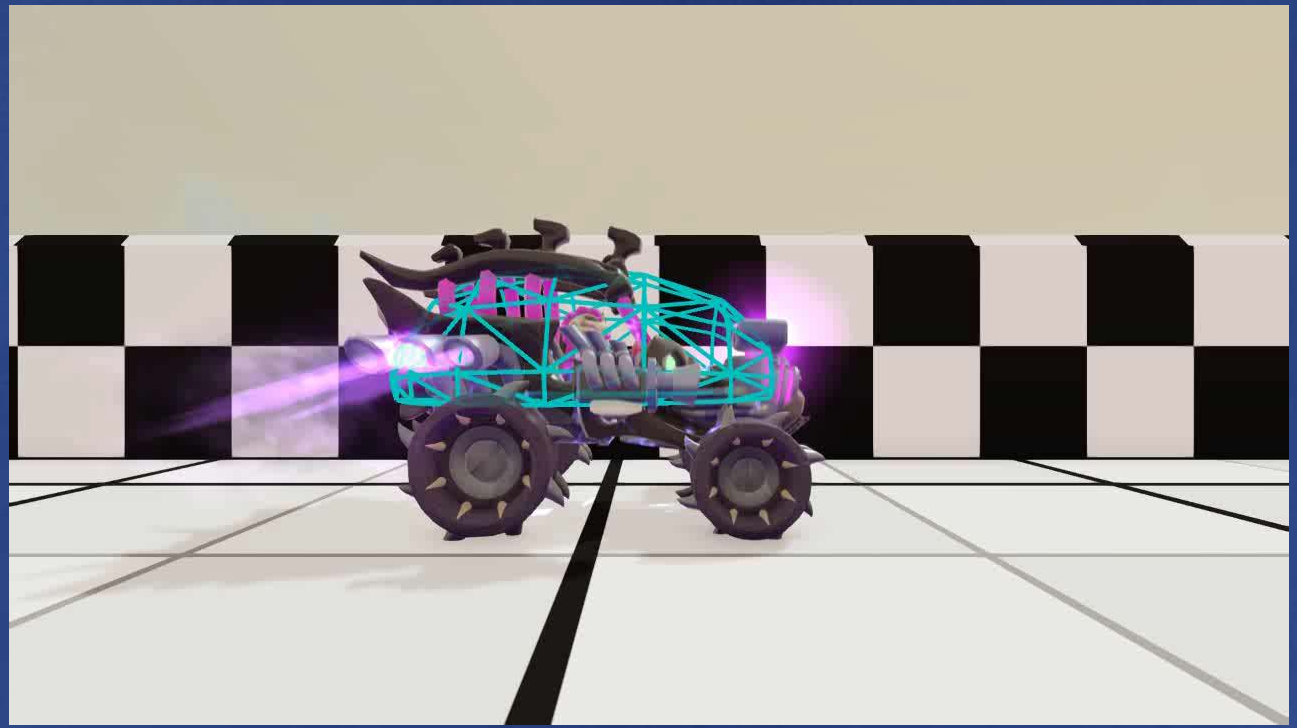
Resting ratio $R = \frac{h_{submerge}}{h_{total}}$

At rest: $F_G = F_B$

$\therefore V = \frac{mass}{R \cdot P}$

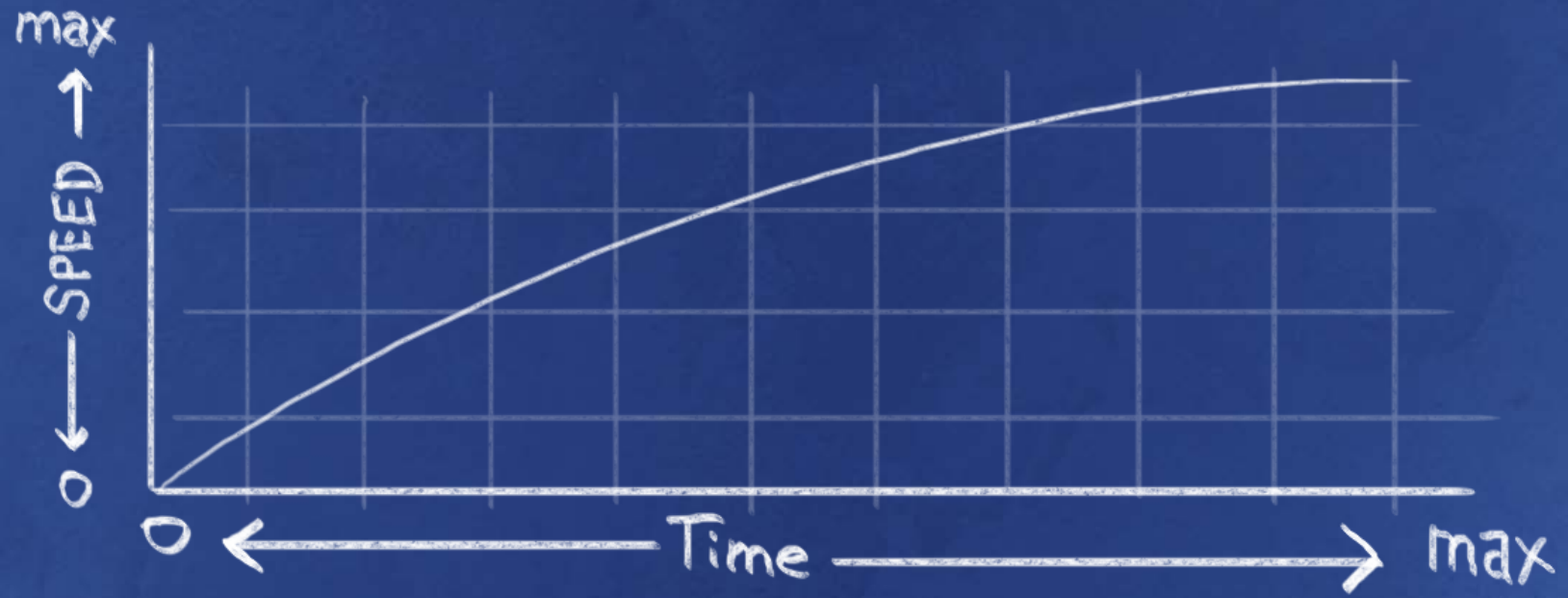
(This means "Therefore")

LAYERED COMPLEXITY THROUGH ANIMATION

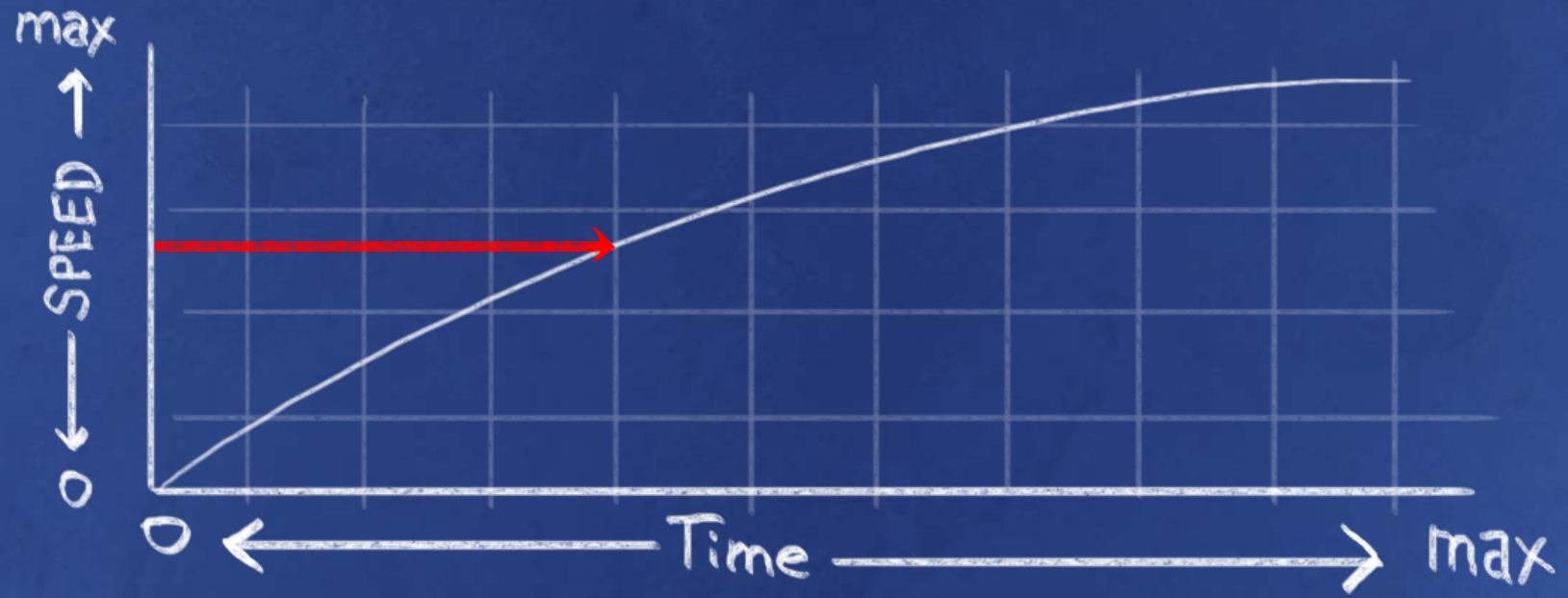


TECHNICAL DEEP DIVE

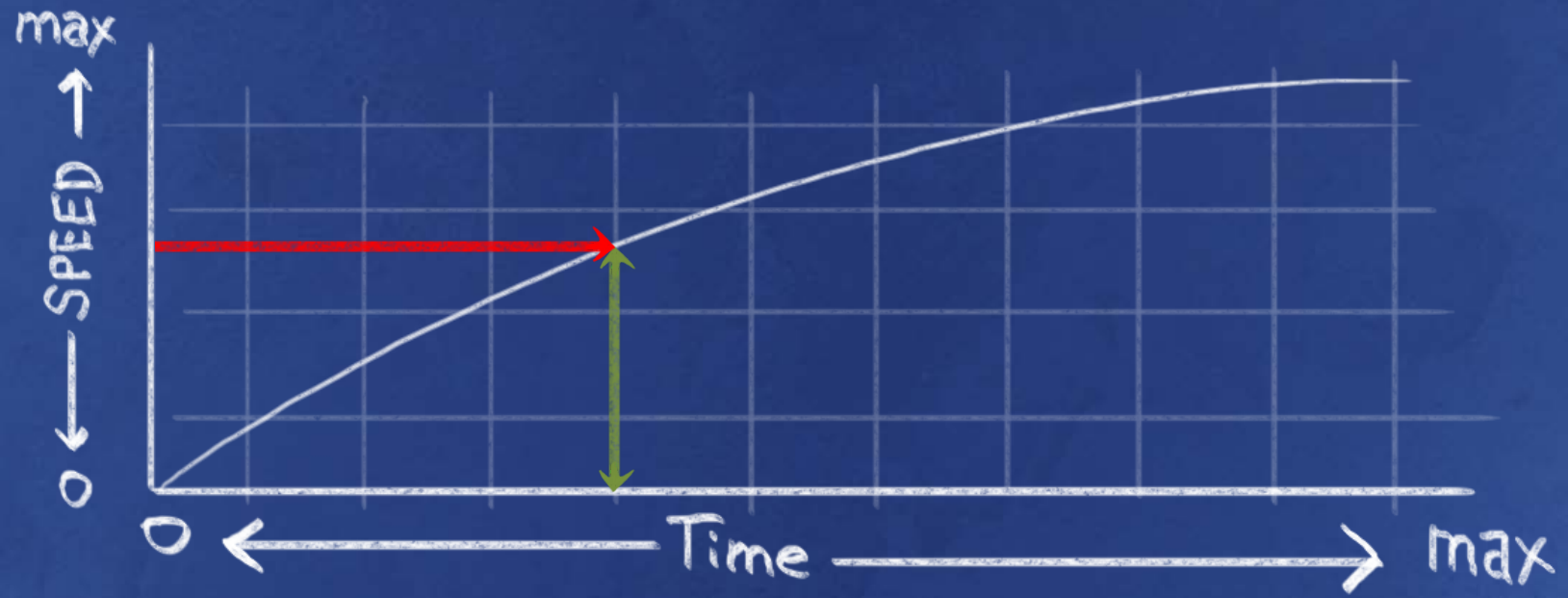
EVALUATING VELOCITY CURVES



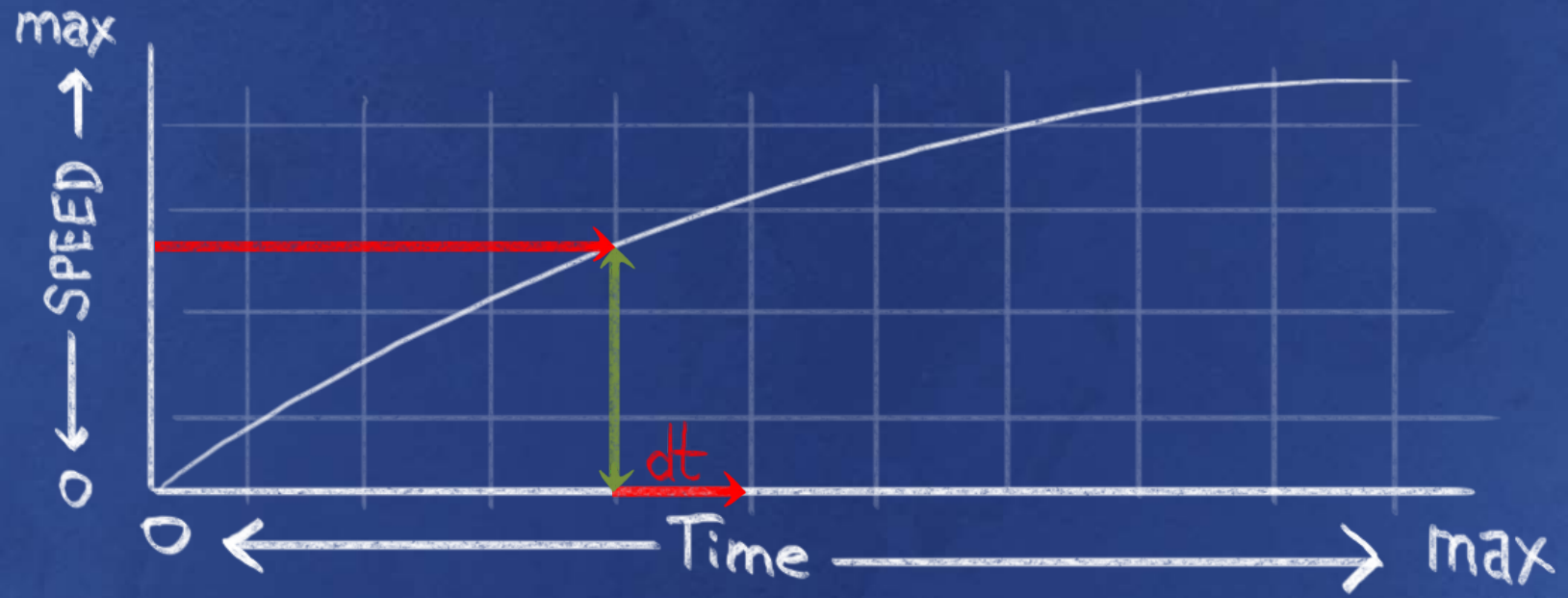
EVALUATING VELOCITY CURVES



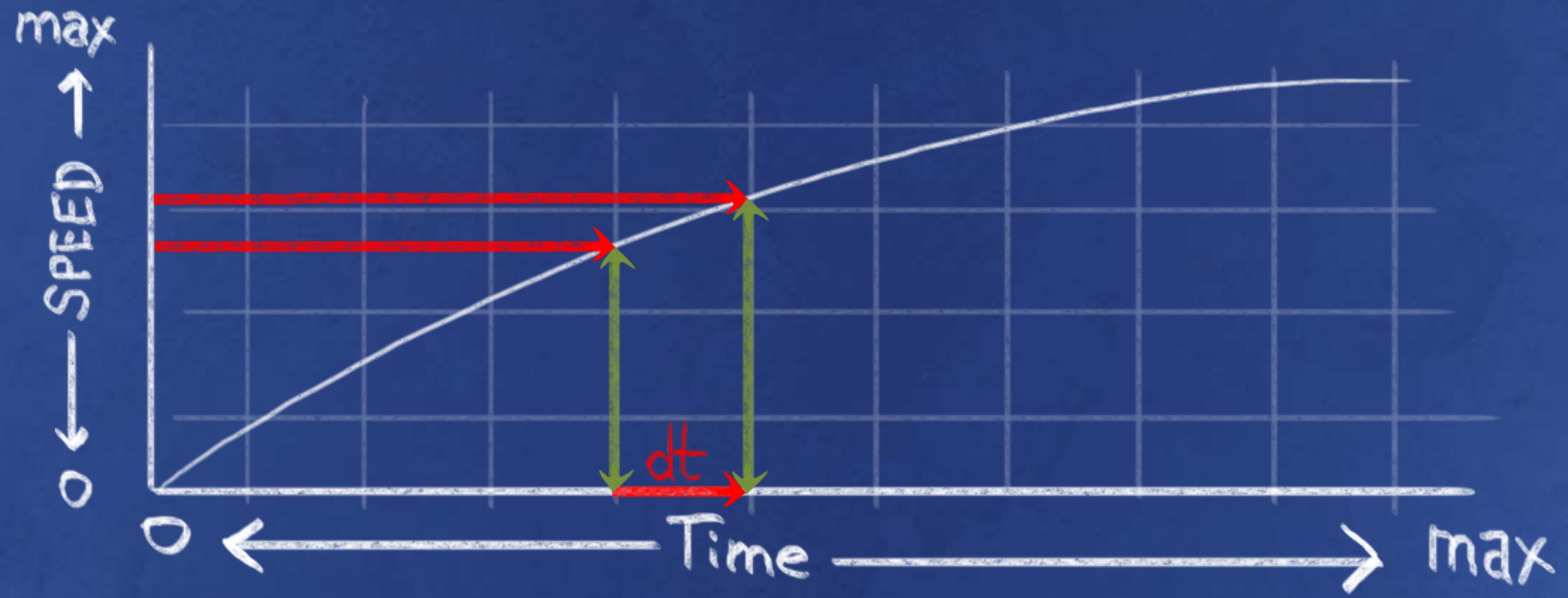
EVALUATING VELOCITY CURVES



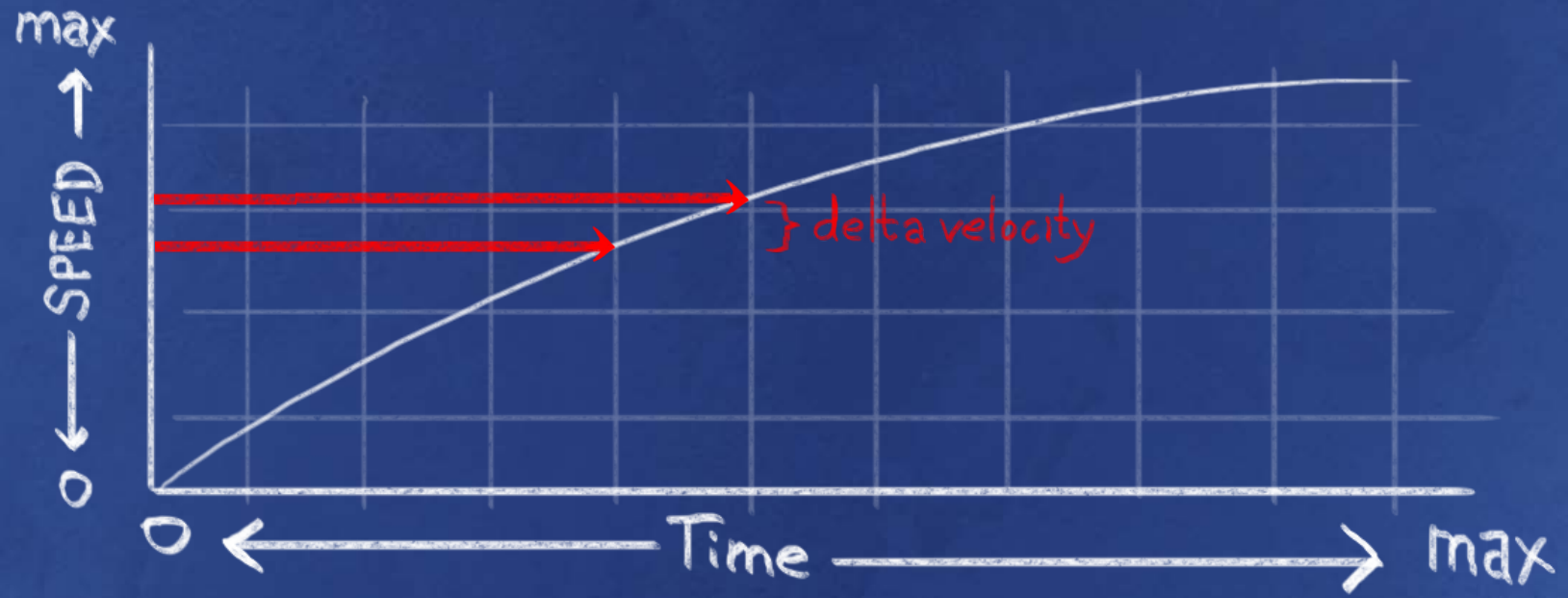
EVALUATING VELOCITY CURVES



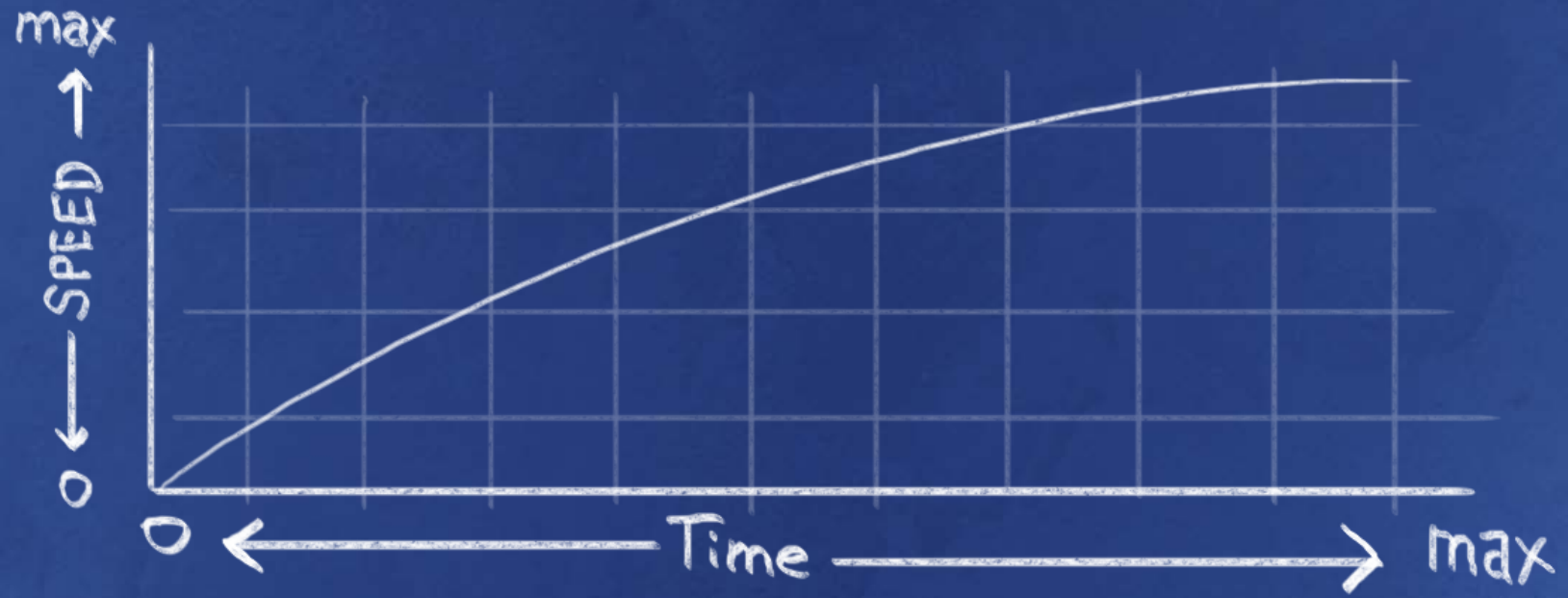
EVALUATING VELOCITY CURVES



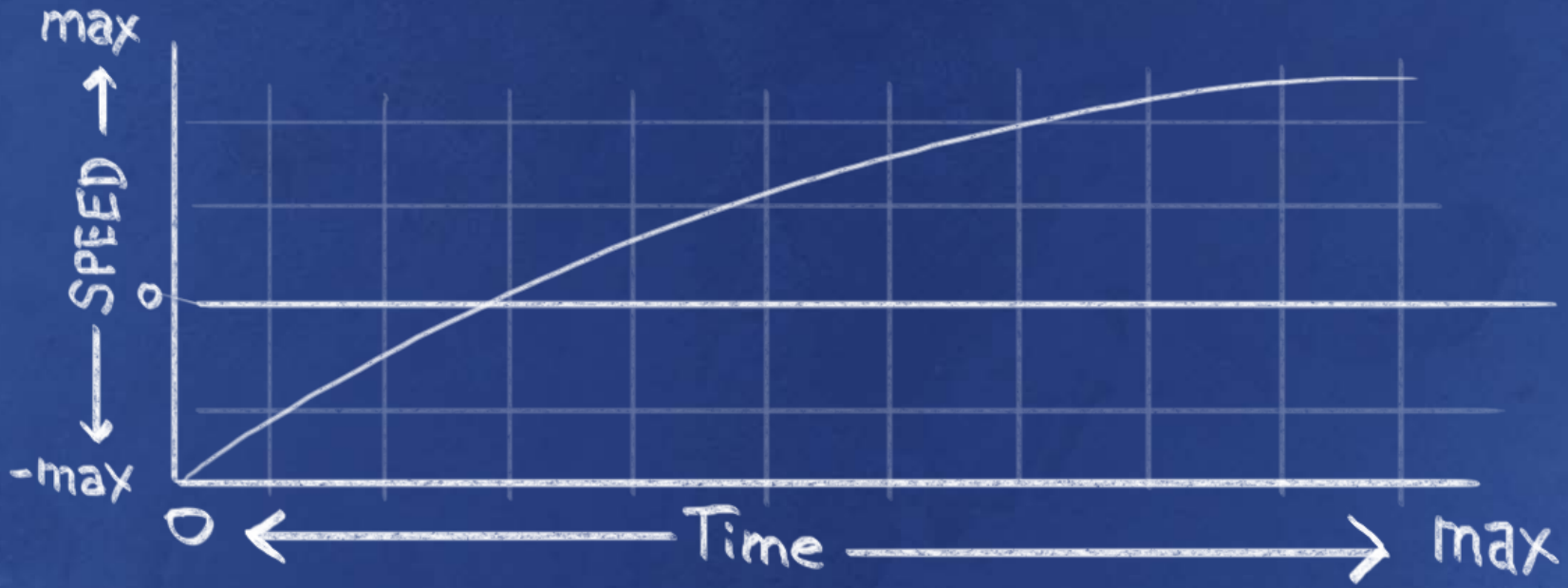
EVALUATING VELOCITY CURVES



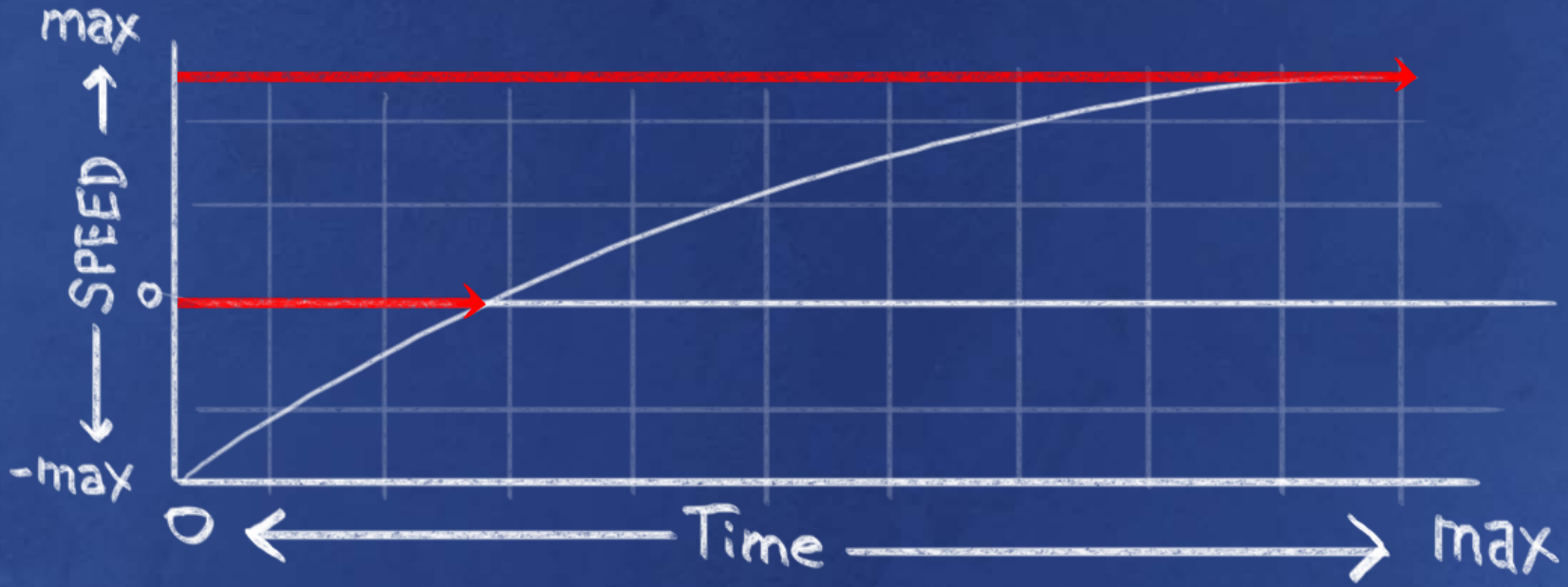
EVALUATING VELOCITY CURVES



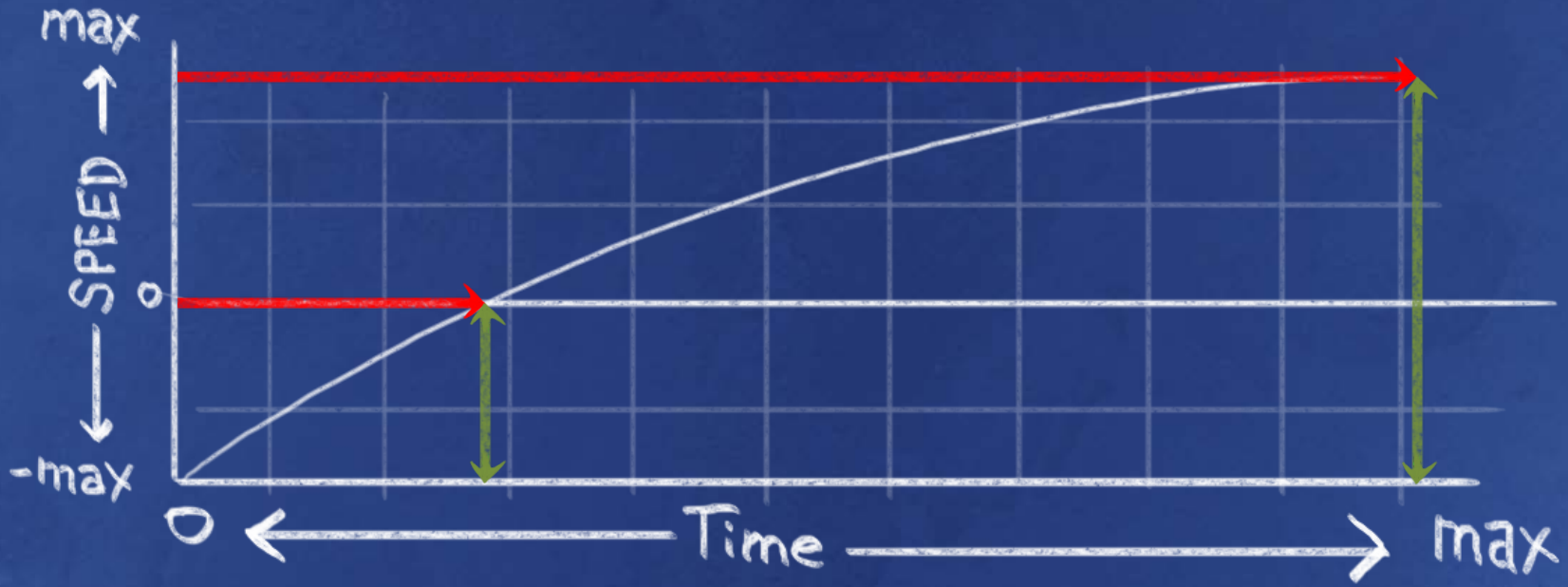
EVALUATING VELOCITY CURVES



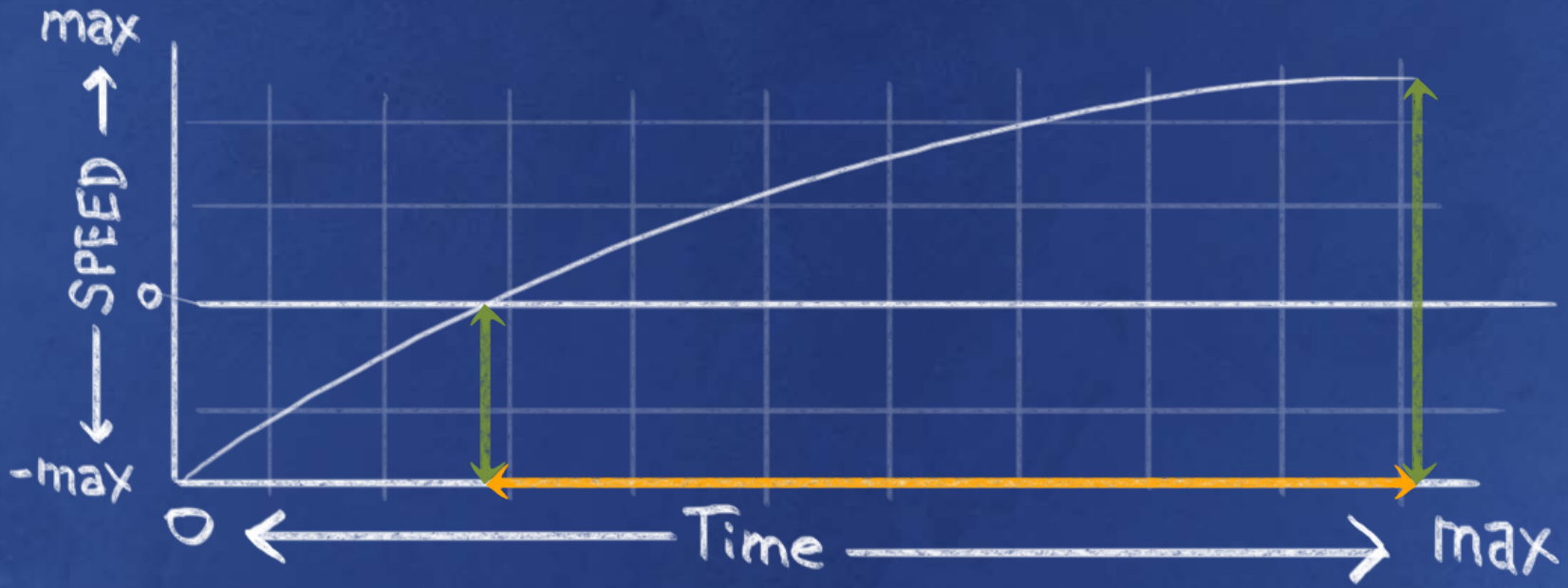
EVALUATING VELOCITY CURVES



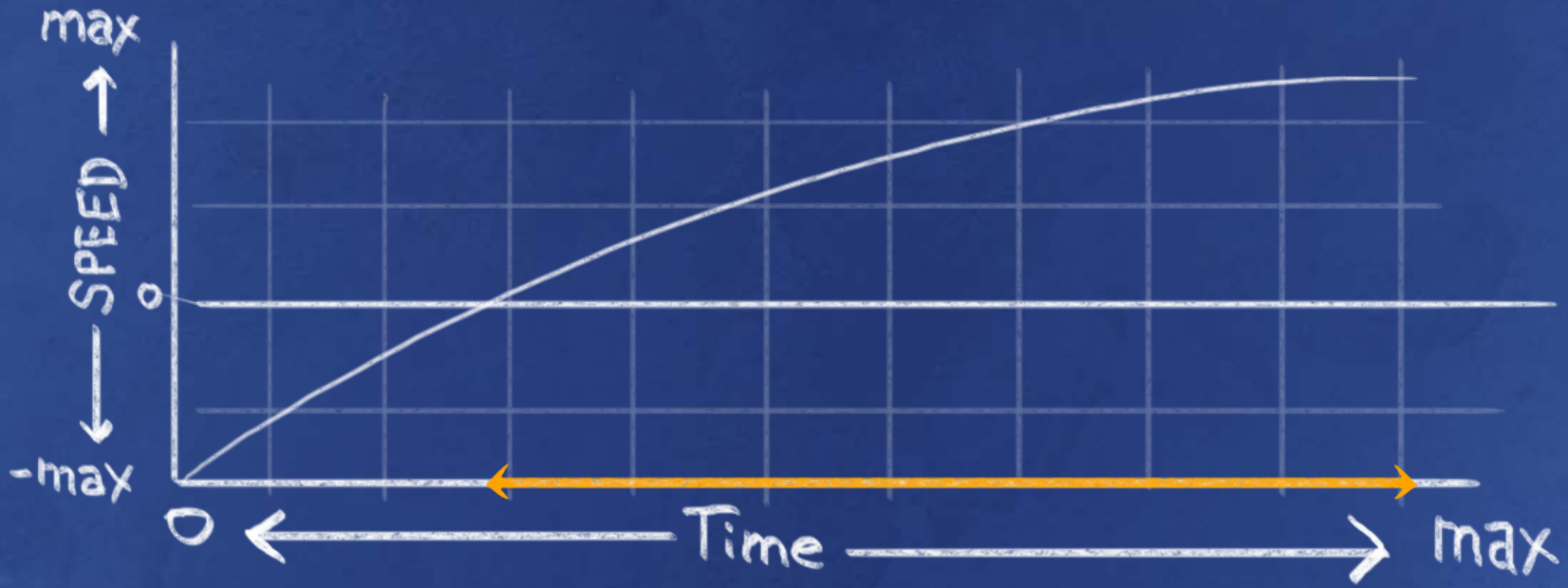
EVALUATING VELOCITY CURVES



EVALUATING VELOCITY CURVES

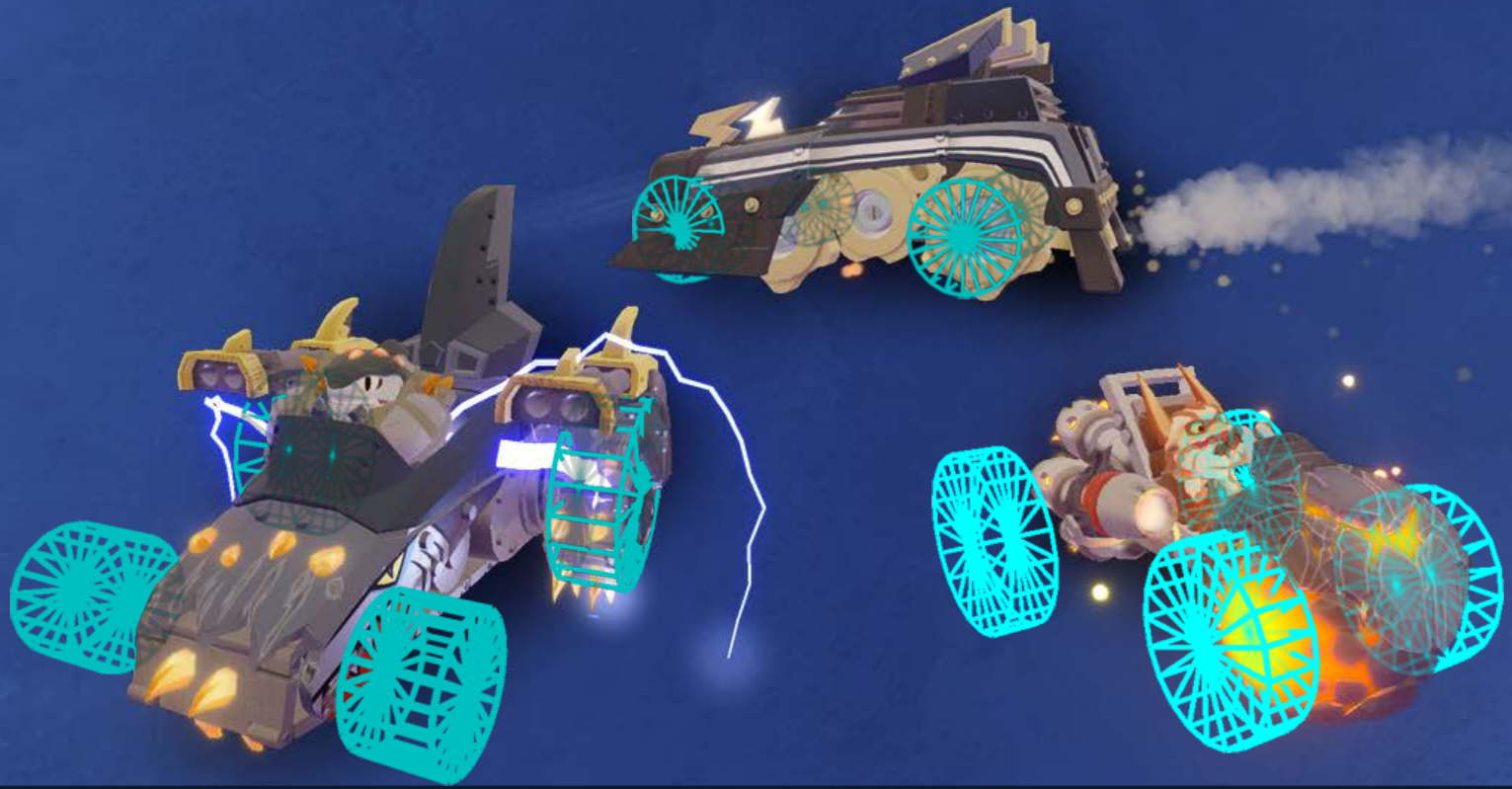


EVALUATING VELOCITY CURVES

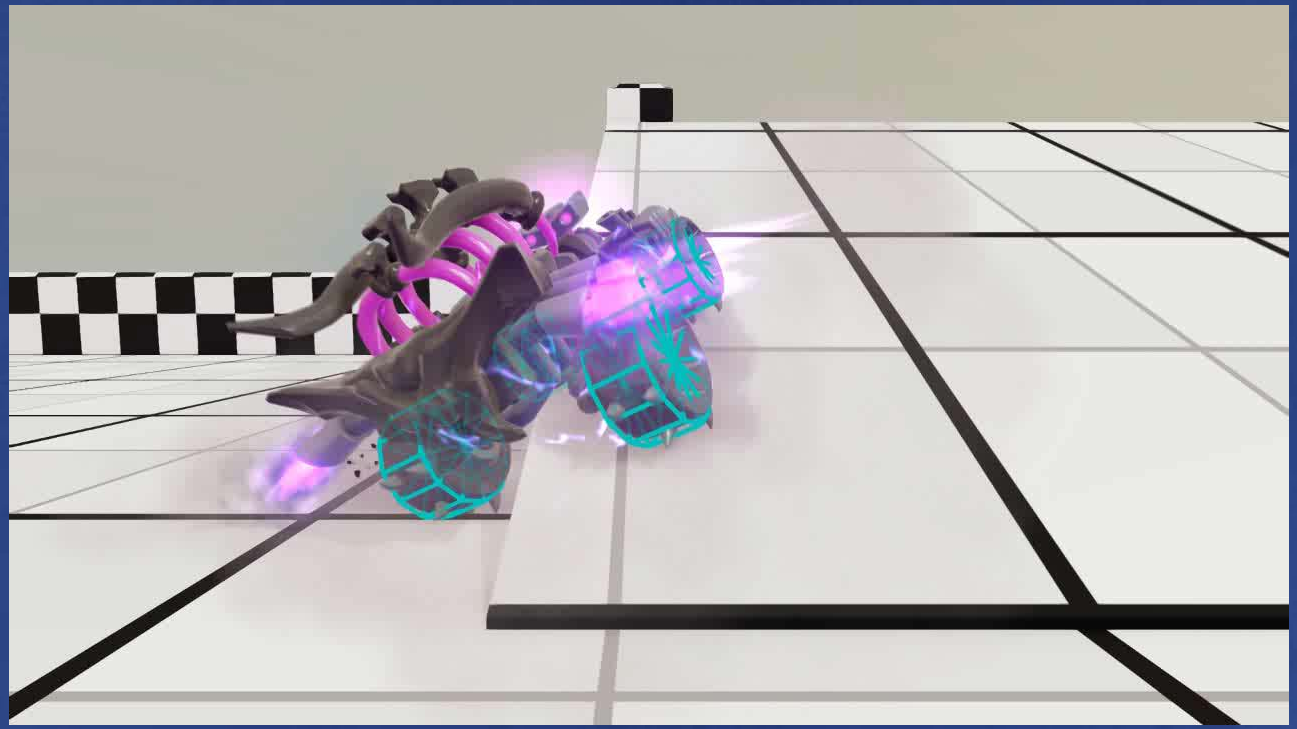


SUSPENSION

SUSPENSION



SHAPE CAST



SHAPE CAST

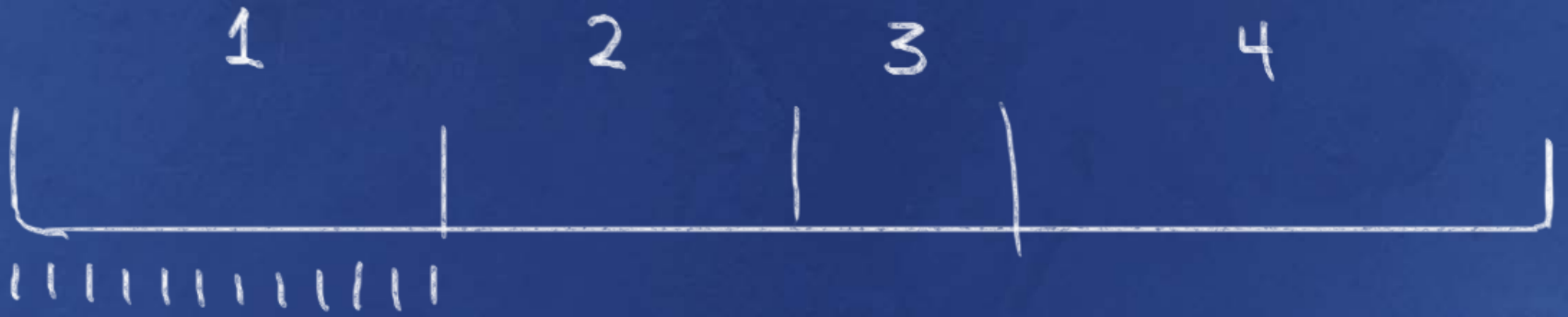
SPRING SIMULATION

SPRING SIMULATION

SPRING SIMULATION

SPRING SIMULATION

SPRING SIMULATION



SPRING SIMULATION

SPRING SIMULATION

SPRING SIMULATION

HIGH-FREQUENCY SUBSTEPPING

```
mTimeToProcess += deltaFrameTime
totalSubStepTime = 0.0f
travelDistance = 0.0f
while (mTimeToProcess >= subStepFixedDeltaTime)
{
    suspVel = rigidBody.getPointVelocity(suspPoint) + gravity * totalSubStepTime
    suspVelProjected = suspDir * suspDir.dot(suspVel)
    springLen = origSpringLen - travelDistance
    // Calculate force based on springLen and suspVelProjected (spring damper)
    totalSubStepTime += subStepFixedDeltaTime
    travelDistance += suspVel Projected* subStepFixedDeltaTime
    mTimeToProcess -= subStepFixedDeltaTime
}
```

HIGH-FREQUENCY SUBSTEPPING

```
mTimeToProcess += deltaFrameTime
totalSubStepTime = 0.0f
travelDistance = 0.0f
while (mTimeToProcess >= subStepFixedDeltaTime)
{
    suspVel = rigidBody.getPointVelocity(suspPoint) + gravity * totalSubStepTime
    suspVelProjected = suspDir * suspDir.dot(suspVel)
    springLen = origSpringLen - travelDistance
    // Calculate force based on springLen and suspVelProjected (spring damper)
    totalSubStepTime += subStepFixedDeltaTime
    travelDistance += suspVel Projected* subStepFixedDeltaTime
    mTimeToProcess -= subStepFixedDeltaTime
}
```

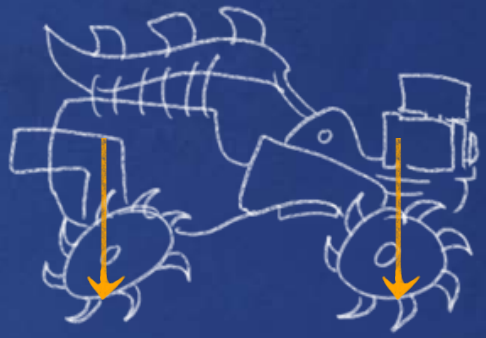
HIGH-FREQUENCY SUBSTEPPING

```
mTimeToProcess += deltaFrameTime
totalSubStepTime = 0.0f
travelDistance = 0.0f
while (mTimeToProcess >= subStepFixedDeltaTime)
{
    suspVel = rigidBody.getPointVelocity(suspPoint) + gravity * totalSubStepTime
    suspVelProjected = suspDir * suspDir.dot(suspVel)
    springLen = origSpringLen - travelDistance
    // Calculate force based on springLen and suspVelProjected (spring damper)
    totalSubStepTime += subStepFixedDeltaTime
    travelDistance += suspVel Projected* subStepFixedDeltaTime
    mTimeToProcess -= subStepFixedDeltaTime
}
```

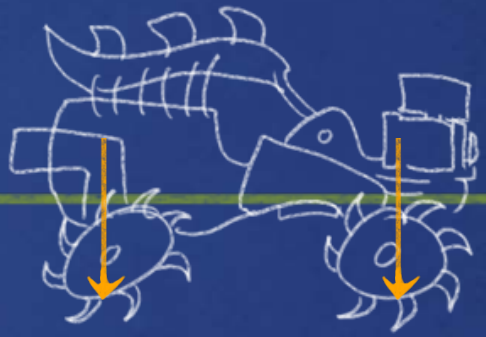
HIGH-FREQUENCY SUBSTEPPING

```
mTimeToProcess += deltaFrameTime
totalSubStepTime = 0.0f
travelDistance = 0.0f
while (mTimeToProcess >= subStepFixedDeltaTime)
{
    suspVel = rigidBody.getPointVelocity(suspPoint) + gravity * totalSubStepTime
    suspVelProjected = suspDir * suspDir.dot(suspVel)
    springLen = origSpringLen - travelDistance
    // Calculate force based on springLen and suspVelProjected (spring damper)
    totalSubStepTime += subStepFixedDeltaTime
    travelDistance += suspVel Projected* subStepFixedDeltaTime
    mTimeToProcess -= subStepFixedDeltaTime
}
```

HIGH-FREQUENCY SUBSTEPPING

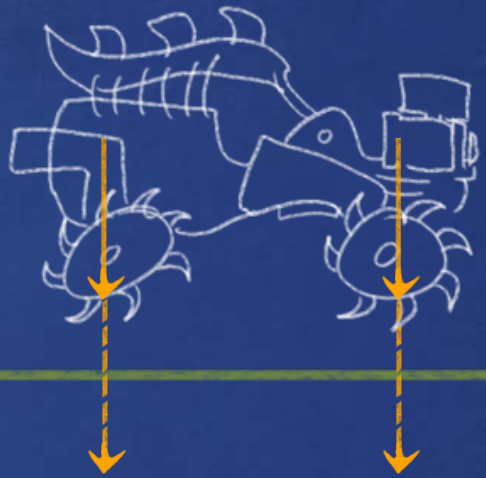


HIGH-FREQUENCY SUBSTEPPING

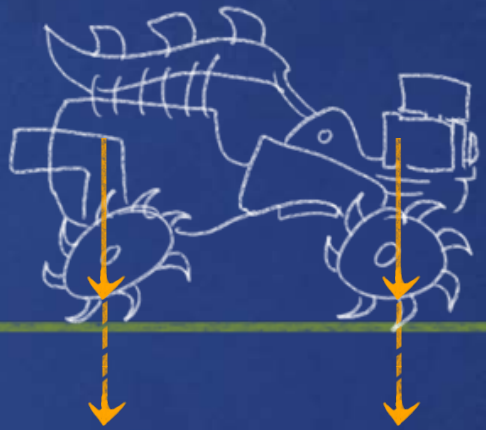


HIGH-FREQUENCY SUBSTEPPING

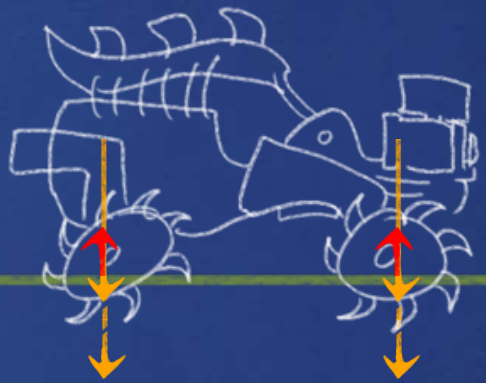
HIGH-FREQUENCY SUBSTEPPING



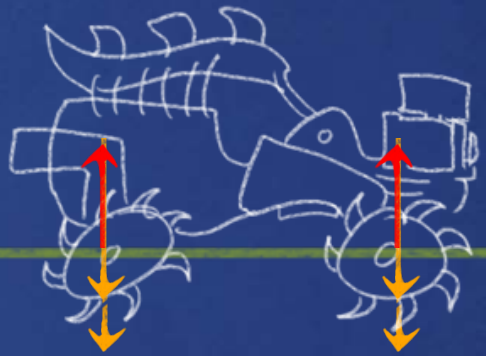
HIGH-FREQUENCY SUBSTEPPING



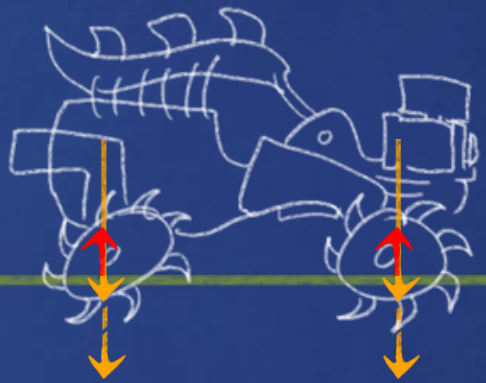
HIGH-FREQUENCY SUBSTEPPING



HIGH-FREQUENCY SUBSTEPPING



HIGH-FREQUENCY SUBSTEPPING



HIGH-FREQUENCY SUBSTEPPING

HIGH-FREQUENCY SUBSTEPPING

COLLISION RESPONSE SYSTEM

*Goal: **Fun** vehicle physics for a young, novice demographic*

Game for kids: expect them to crash into things

Flexible one-stop shop for all collision responses involving vehicles

- *Had to work for all vehicle types*

Data driven

COLLISION RESPONSE SYSTEM

COLLISION RESPONSE SYSTEM

Rule based

COLLISION RESPONSE SYSTEM

Rule based

- *Set of criteria*

COLLISION RESPONSE SYSTEM

Rule based

- *Set of criteria*
- *Priority*

COLLISION RESPONSE SYSTEM

Rule based

- *Set of criteria*
- *Priority*
- *Must involve a vehicle*

COLLISION RESPONSE SYSTEM

Rule based

- *Set of criteria*
- *Priority*
- *Must involve a vehicle*
- *Scoped*

COLLISION RESPONSE SYSTEM

Rule based

- *Set of criteria*
- *Priority*
- *Must involve a vehicle*
- *Scoped*
- *Bi-directional*

COLLISION RESPONSE SYSTEM

Rule based

- *Set of criteria*
- *Priority*
- *Must involve a vehicle*
- *Scoped*
- *Bi-directional*
- *Response*

COLLISION RESPONSE SYSTEM

COLLISION RESPONSE SYSTEM

Phase 1: Mid-simulation

- *Limited*
- *Predictive*

COLLISION RESPONSE SYSTEM

Phase 1: Mid-simulation

- *Limited*
- *Predictive*

Phase 2: Post-solve

- *Unlimited*
- *Reactive*

COLLISION RESPONSE SYSTEM

Phase 1: Mid-simulation

- *Limited*
- *Predictive*
- *Best match per contact*

Phase 2: Post-solve

- *Unlimited*
- *Reactive*

COLLISION RESPONSE SYSTEM

Phase 1: Mid-simulation

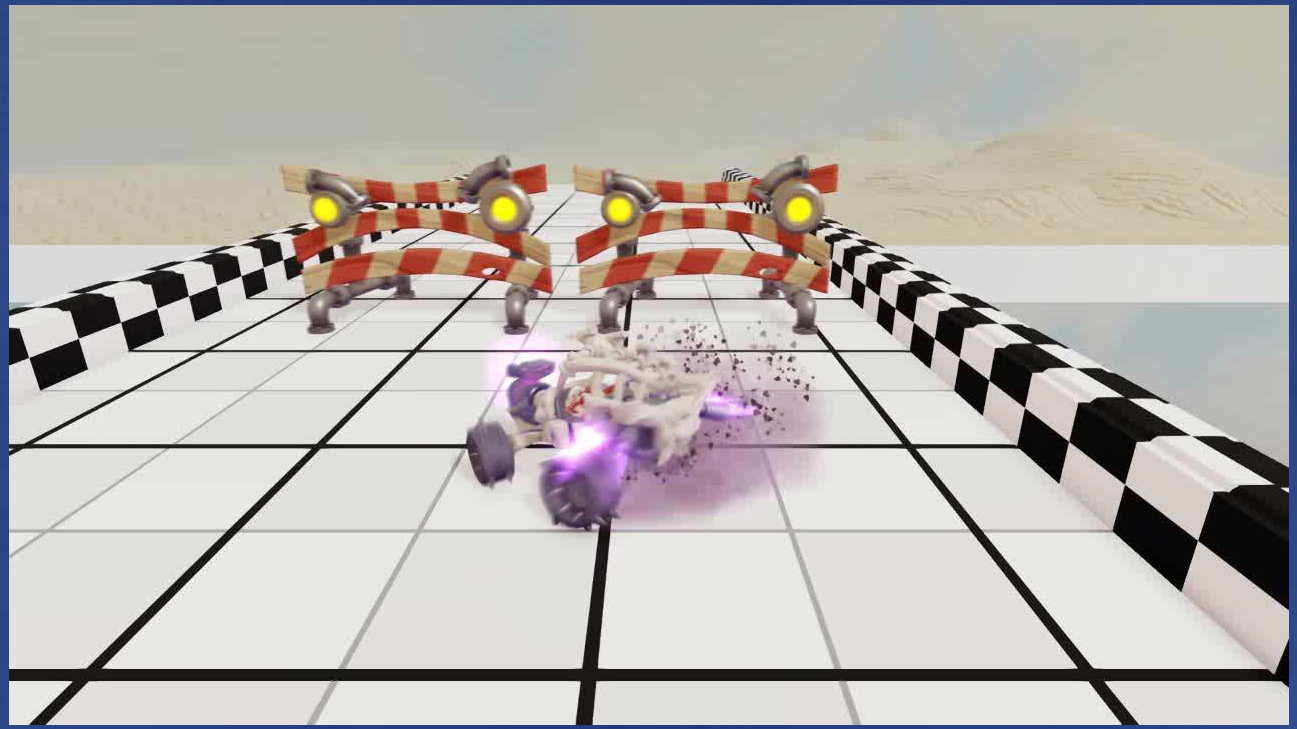
- *Limited*
- *Predictive*
- *Best match per contact*

Phase 2: Post-solve

- *Unlimited*
- *Reactive*
- *Best match between body pairs*

COLLISION RESPONSE SYSTEM

COLLISION RESPONSE SYSTEM



COLLISION RESPONSE SYSTEM

	SOURCE	TARGET
Criteria		
Reaction		

COLLISION RESPONSE SYSTEM

	SOURCE	TARGET
Criteria	Is Vehicle = True	
Reaction		

COLLISION RESPONSE SYSTEM

	SOURCE	TARGET
Criteria	Is Vehicle = True Min Speed = 20%	
Reaction		

COLLISION RESPONSE SYSTEM

	SOURCE	TARGET
Criteria	Is Vehicle = True Min Speed = 20%	
Reaction	Ignore Collision	

COLLISION RESPONSE SYSTEM

	SOURCE	TARGET
Criteria	Is Vehicle = True Min Speed = 20%	
Reaction	Ignore Collision	Apply Damage

COLLISION RESPONSE SYSTEM

COLLISION RESPONSE SYSTEM

Be prepared for complexity

- *Layering rules is hard*

COLLISION RESPONSE SYSTEM

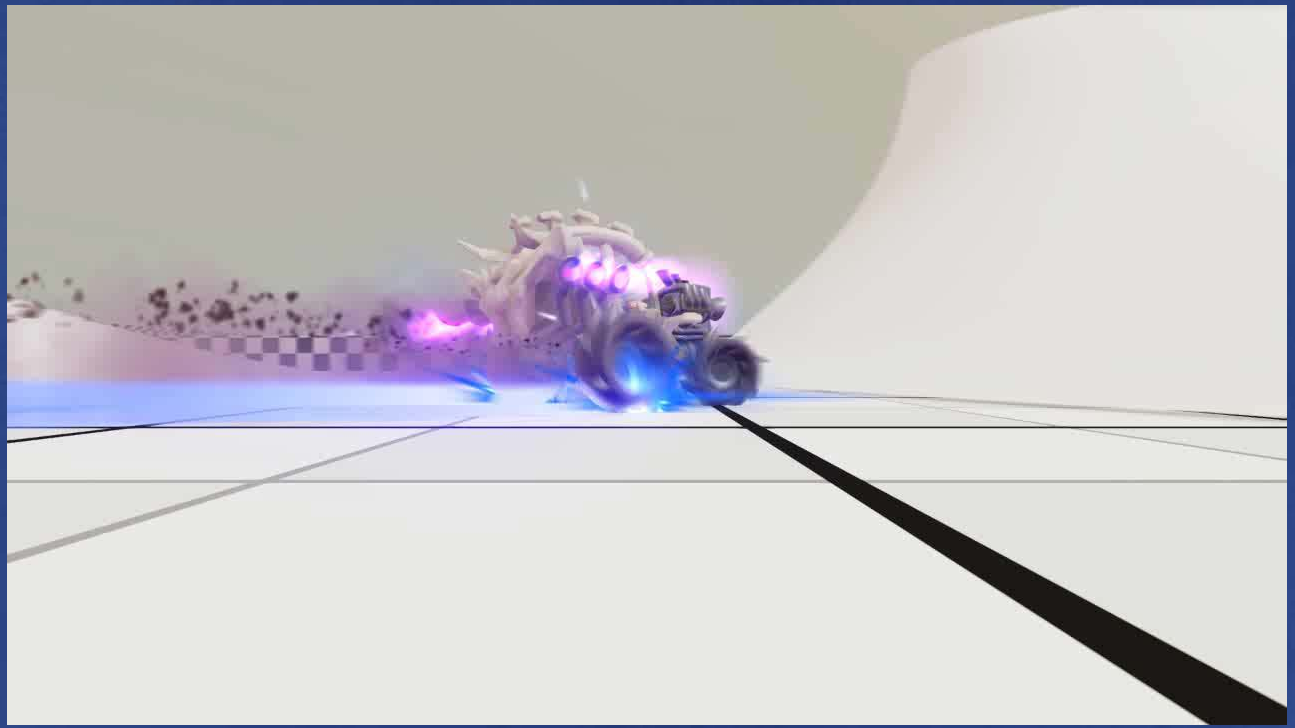
Be prepared for complexity

- *Layering rules is hard*

Flexibility is key

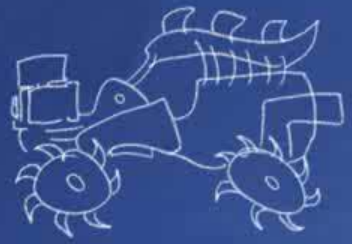
- *Scope of rules*
- *Criteria and response capabilities*

PREDICTIVE LANDING



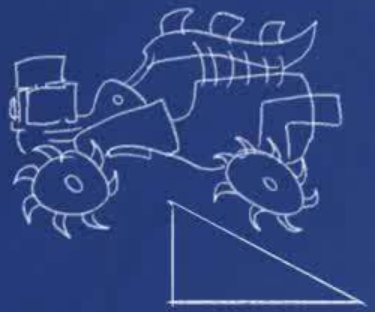
PREDICTIVE LANDING

PREDICTIVE LANDING



PREDICTIVE LANDING

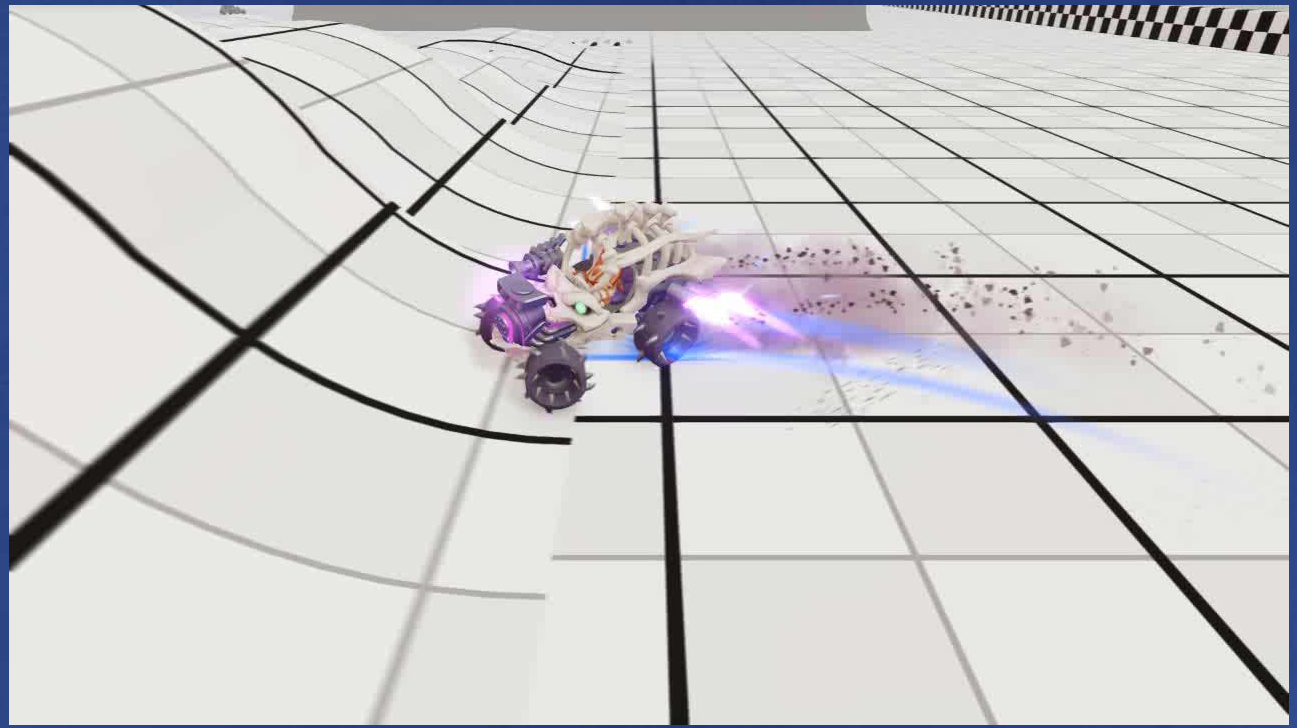
PREDICTIVE LANDING



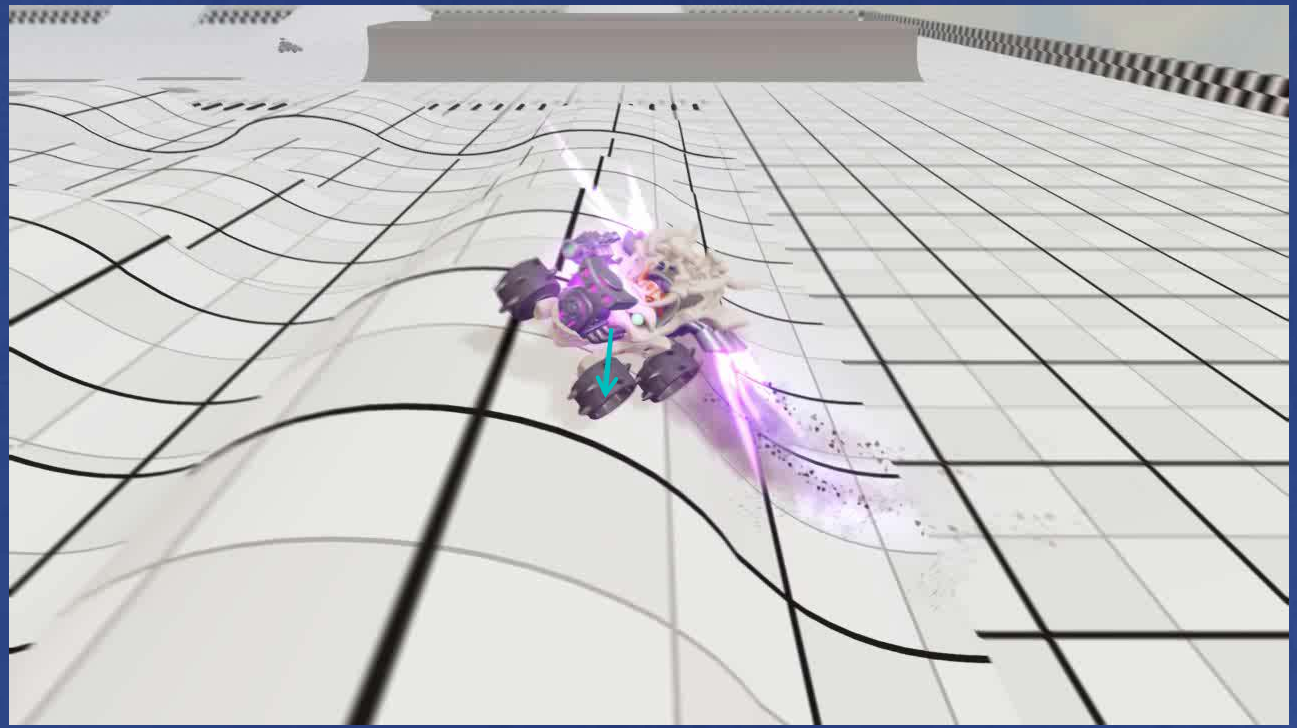
PREDICTIVE LANDING

PREDICTIVE LANDING

PREDICTIVE LANDING



PREDICTIVE LANDING



CONCLUSION

QUESTIONS?

Contact Info:

Jan-Erik: jsteel@vvisions.com

Patrick: pdonnelly@vvisions.com

Thanks to Scott Moore for lending his artistic talents to this talk, Chris Butcher for mentoring us and the team at VV for making all this possible.