Motion Matching

road to
next-gen animation

Simon Clavet
Ubisoft Montreal
Who is this guy
animation programmer at Ubisoft Montreal since 2005
Animation Goals

Precise Gameplay

Believable Animation
This Talk:
- Little history of animation systems
- Motion Matching
- Workflow
- Procedural Touchups
On the first day,

God created the function PlayAnim()
PlayAnim(RunAnimation);
// start
if (!walking && wantToWalk)
{
    PlayAnim(StartAnim);
    walking = true;
}

// walk loop
if (IsPlaying(StartAnim) && IsAtEndOfAnim())
{
    PlayAnim(WalkLoopAnim);
}

// stop
if (walking && !wantToWalk)
{
    PlayAnim(StopAnim);
    walking = false;
}
State Machines
if (speed > 3.0f)
{
    PlayAnim(RunAnim);
}
else if (speed > 0.0f)
{
    PlayAnim(WalkAnim);
}
else
{
    PlayAnim(IdleAnim);
}
Blend/Decision Trees
Question 1
Where would you put this animation in your structure?

Start-Strafe90-TurnOnSpot45-Stop
Parametric Blends

- speed
- slope
- strafe angle
// a parametric-blend query
AnimQuery query;

// let's strafe up a slope
query.AddDesiredFeature("speed", 3.0f);
query.AddDesiredFeature("slope", 35.0f);
query.AddDesiredFeature("strafeAngle", 90.0f);

Array<float> blendFactors = ComputeBlendFactors(query);

SetBlendFactors(blendFactors);
Question 2
Is there a way to deal with loops and transitions in a uniform way?
Unstructured list of animations

Preprocess:
Figure out *when you can* jump to somewhere else

Motion Graphs

Lucas Kovar
University of Wisconsin-Madison

Michael Gleicher*
University of Wisconsin-Madison

Frédéric Pighin†
University of Southern California
Institute for Creative Technologies

Motion Graphs

- Figure out when you can jump to somewhere else

is desired, then often there is little that can be done, more data, a time-consuming and expensive problem. In particular, is a problem for applications that require...
When reaching a transition point, decide where to go from a list of possibilities
A path on the graph gives us our animation
Question 3
How do we choose the next animation?
Reinforcement Learning based Character Locomotion in Hitman: Absolution

Michael Büttner
Technical Director - IO Interactive / Square Enix
For comfortable control, we need a dense graph
Motion Fields for Interactive Character Control

Yongjoon Lee\textsuperscript{1,2,*} Kevin Wampler\textsuperscript{1†} Gilbert C. H. Lo\textsuperscript{1}
\textsuperscript{1}University of Washington

Abstract

We propose a novel representation of motion data and control which gives characters highly agile responses to user input and allows a natural handling of arbitrary external disturbances. Our representation organizes samples of motion data into a high-dimensional generalization of a vector field which we call a motion field. Our run-time motion synthesis mechanism freely flows through the motion field in response to user commands.

Figure 2: Action search using a value function.

Our representation organizes samples of motion data into a high-dimensional generalization of a vector field which we call a motion field. Our run-time motion synthesis mechanism freely flows through the motion field in response to user commands.

1 Introduction

Human motion is a highly-varied and continuous phenomenon: it
The problem with Motion Fields:

the equations are scary...
Let’s just use the main idea of the paper:

We can jump to any other frame whenever we want
Question 4
How to choose the ‘correct’ start time in a stop animation?

pose matching?
velocity matching?
precise end position matching?
a mix of all of the above?
Introducing *Motion Matching*
A ridiculously brute-force approach to animation selection
Algorithm:

Every frame, look at all mocap and jump at the best place.
Every candidate jumping point has a cost.

If the candidate matches the current situation and the piece of motion that follows brings us where we want, then the cost is zero.
(Style: very tired...)

Switch multiple times per second
Step 1: Mocap
Step 2: Code
int m_CurrentAnimIndex;
float m_CurrentAnimTime;

void AmoUpdate(Goal goal, float dt)
{
    m_CurrentAnimTime += dt;
    Pose currentPose = EvaluateLerpedPoseFromData(m_CurrentAnimIndex, m_CurrentAnimTime);

    float bestCost = 1000000;
    Pose bestPose;

    // loop on all mocap
    for (int i = 0; i < m_Poses.Size(); i++)
    {
        Pose candidatePose = m_Poses[i];

        // every candidate jumping point has a cost
        float thisCost = ComputeCost(currentPose, candidatePose, goal);
        if (thisCost < bestCost)
        {
            // remember the best candidate
            bestCost = thisCost;
            bestPose = candidatePose;
        }
    }
}
bool theWinnerIsAtTheSameLocation =
    m_CurrentAnimIndex == bestPose.m_AnimIndex &&
    fabs(m_CurrentAnimTime - bestPose.m_AnimTime) < 0.2f;

if (!theWinnerIsAtTheSameLocation)
{
    // blend to the winning location
    m_CurrentAnimIndex = bestPose.m_AnimIndex;
    m_CurrentAnimTime = bestPose.m_AnimTime;
    PlayAnimStartingAtTime(m_CurrentAnimIndex, m_CurrentAnimTime, 0.25f);
}

float ComputeCost(Pose currentPose, Pose candidatePose, Goal goal)
{
    float cost = 0.0f;

    // how much the candidate jumping position matches the current situation
    cost += ComputeCurrentCost(currentPose, candidatePose);

    return cost;
}
Trick 1: Posematch only a few bones
Precompute and save with the animation for fast cost computation

Pose/Velocity Matching

- Local velocity
- Feet positions
- Feet velocities
- Weapon position
- Etc.
How to follow a desired trajectory?
Trick 2: Just check where a piece of animation brings you if you play it.
class TrajectoryPoint
{
    Vector3 m_Position
    float m_Sight;
    float m_TimeDelay;
};

// desired goal, sent by gameplay each frame
class Goal
{
    Array<TrajectoryPoint> m_DesiredTrajectory;

    Stance m_DesiredStance;

    // ...
};
Trajectory Matching

- Future position
- Future orientation
- Future velocity/acceleration

**cost**
float ComputeCost(Pose currentPose, Pose candidatePose, Goal goal)
{
    float cost = 0.0f;

    // how much the candidate jumping position matches the current situation
    cost += ComputeCurrentCost(currentPose, candidatePose);

    // this is our responsivity slider
    static float responsivity = 1.0f;

    // how much the candidate piece of motion matches the desired trajectory
    cost += responsivity * ComputeFutureCost(candidatePose, goal);

    return cost;
}
Now we can match more things
Future Stance Matching

Left Stance

Right Stance

Candidate

Predicted future pose for this candidate

Raw Mocap
Elegantly find transitions when they exist
Optimizations  (subject of a future talk)

- **LOD**
- **KD-Tree**
- **Motion Shaders**
- **Etc.**
Trajectory Simulation Choices

- *Displacement from Animation?*
- *Displacement from Simulation?*
Our choice:

The code decides the real simulation point
Trajectory Simulation

Spring damper on velocity: predictable and comfortable
Choose animations that follow the simulation

And do *Displacement from Animation*
Animation moves and rotates the entity.
Clamp the entity 15 cm around the simulation.
Future Trajectory Prediction

- Run actual simulation code to predict future trajectory
- Predict stops on walls and ledges
Future Trajectory Prediction
● Little history of animation systems
● Motion Matching
● Workflow
● Procedural Touchups
Mocap is tweaked, imported, and marked up.

At runtime Gameplay makes a *request* (desired trajectory and event constraints).

The animation system continuously finds the best piece of data to play.

We modify the result to precisely match gameplay and environment.

Pipeline:

- **Gameplay Code and Logic State-Machine**
- **request**
- **Pose/Trajectory/Event Matching**
- **anim**
- **Procedural Warping**
Simple timed state-machine
Possible to play the game without animation! (just debug display)
Events are placed at the *most significant* moment in the move, as determined by convention.
The clip says: *I want this event to happen exactly 0.8 seconds after I start*
Event Setup

Event have a type, which will get requested by the clip

Event constraints matched in gameplay goal
Movement
Little history of animation systems
Motion Matching
Workflow
Procedural Touchups
Orientation corrections

- Let the anim decide rotation but
- Correct to match future desired position
- Or to match future desired orientation
Procedural Rotation

Without Rotation Correction
Procedural Rotation

Rotate over time to match the desired future position
Procedural Rotation
Procedural Rotation

When future orientation is more important than future position, rotate entity to match that instead
Procedural Rotation
Procedural Rotation
What if Gameplay wants to double the speed?
Timescale the animation according to error ratio

Predicted Error
Just sliding often gives better results
Sliding Prevention

Lots of sliding from:
- Blending very often
- Keeping up with gameplay
Lock the toe when it *doesn’t move too much* in the main animation.
Slope Warping

Carefully tweak ground smoothing
Smoothly pull the hips down
Slope Warping

Prefer smoothness over absolute penetration prevention

Don’t break the pose:
*Never* hyper-extend the knee
Slope animations are automatically chosen by 3D trajectory matching.

IK only compensate for what is missing from the anim.
Spine Pitch Bending

Need precise interactions on any terrain

Sword IK?

Not general enough
Spine Pitch Bending

Keep whole upper body in sync
All together now
Future Work

- Match more stuff
  - Surfaces
  - Interaction Partners
Conclusion

- Motion Matching is a simple idea, that helps us reason about movement description and control.
Conclusion

- It’s also a new type of animation system, with three advantages:
  - High quality
  - Controllable responsiveness
  - Minimal manual work
Final Shower Thought
Let’s just markup the mocap with the *inputs* that should trigger the moves. And generate everything automatically...
Questions?

Thanks to
Jonathan Bapst
Michael Buttner
Sebastien Comte
Serge Doré
Sean Gomez
Xavier Guilbeault
Yves Jacquier
Dany Joannette
Karim Kochen
Marc-Antoine Lamarche
Xavier Lemaire
James Michel
Pascal Mimeault
Khai Nguyen
Guillaume Picard
Alexandre Pichette
Jason VandenBerghe
all For Honor team
and all mocap actors...

More Questions?
Meet me at the Ubisoft Lounge
West Hall 2nd floor
from 2PM to 3PM