Animation System Layers

- Animation Pipeline
- Action state Machine
- Animation controllers

Animation System Layers

- Animation Pipeline
  - Takes animation clips and blends
  - Calculates global pose for the skeleton
  - Calculates matrix palette
  - IK is performed here
- Action state Machine
  - Actions of a character are modelled via a finite state machine
  - Sits atop animation pipeline
  - Provides a state driven animation interface
  - Transitions and transplanting
- Animation controllers
  - Behaviour control
  - Custom tailored to manage the character's behaviour
  - Fighting, driving, climbing
  - Top level player control or AI logic uncluttered by animation micromanagement

The Animation Pipeline

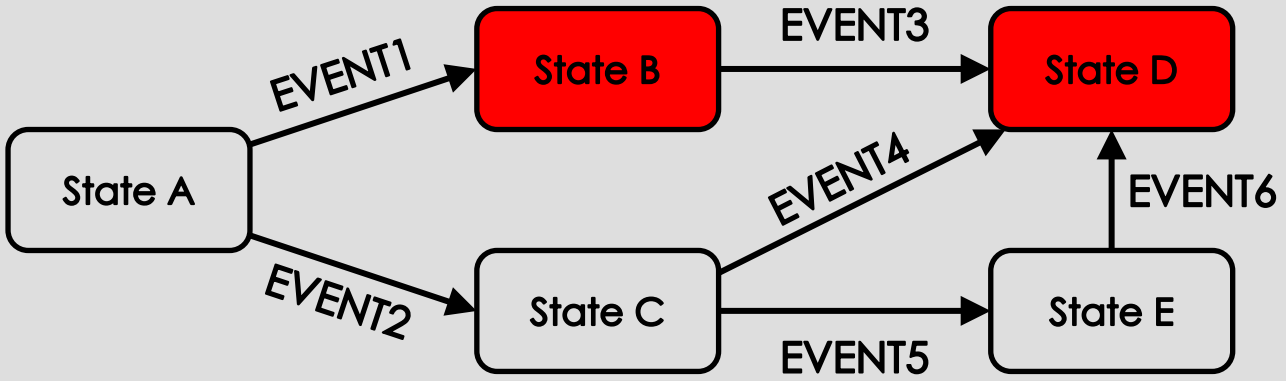
1. Clip decompression & pose extraction
2. Pose Blending
3. Global Pose Generation
4. Post-Processing
5. Global Pose re-Generation
6. Matrix Palette Generation

Action State Machine

- Sits on top of the animation pipeline
  - Actions of the characters are controlled in a state-driven manner
  - Actions of a character are modelled via a finite state machine
- Responsible for smooth transitions
- Can also be used for transplanting

State Machines

- We will define the state machine as a connected graph of states and transitions
- At any time, exactly one of the states is the current state
- Transitions are assumed to happen instantaneously



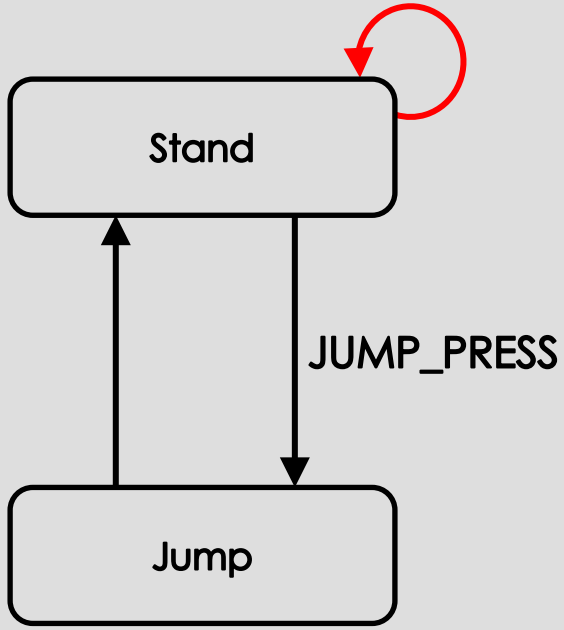
State Machines: Events

- In the context of animation sequencing, we think of
  - States as representing individual animation clips
  - Transitions as being triggered by some sort of event

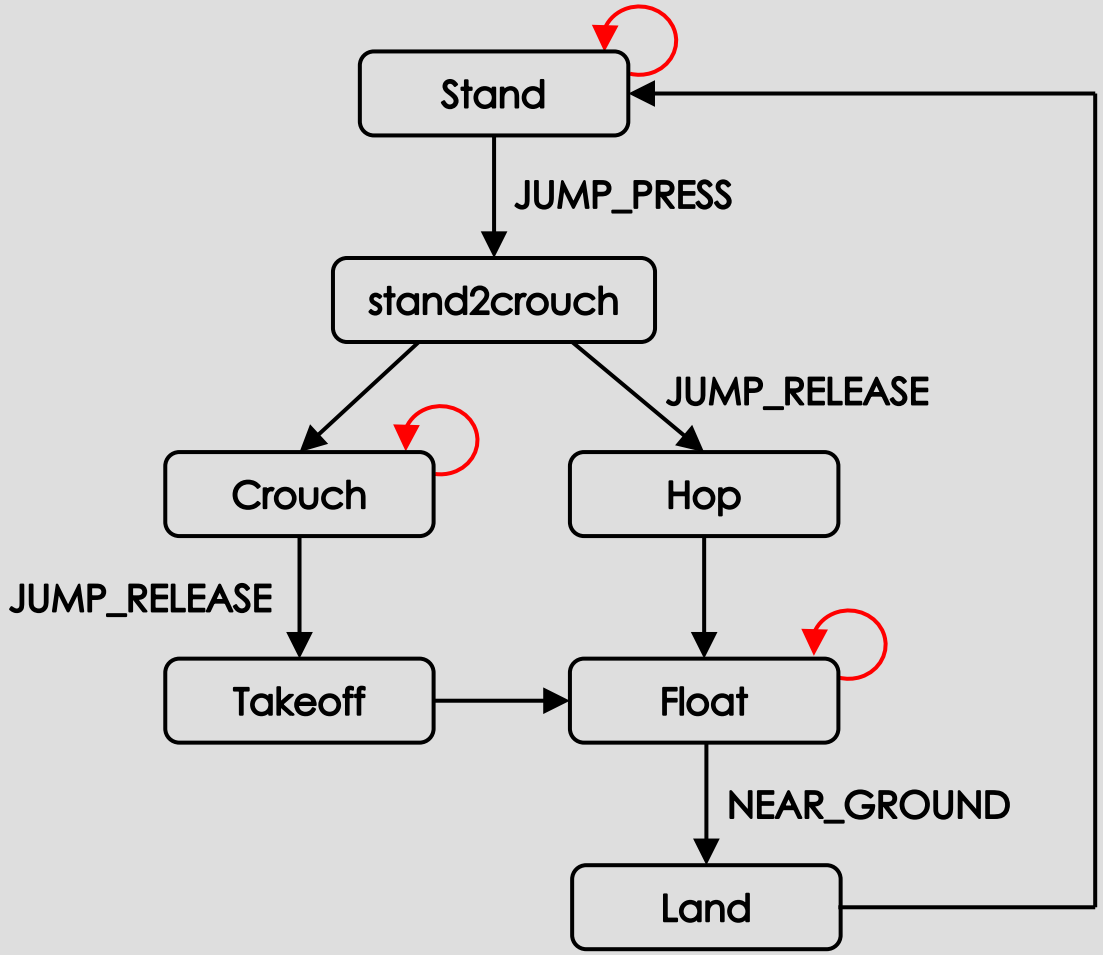
- An event might come from some internal logic or some external input (button press…)

Simple Jump State Machine

- Consider this simple state machine
  - Character jumps upon receiving a JUMP_PRESS message



More Complex Jump



State Machine (Text Version)

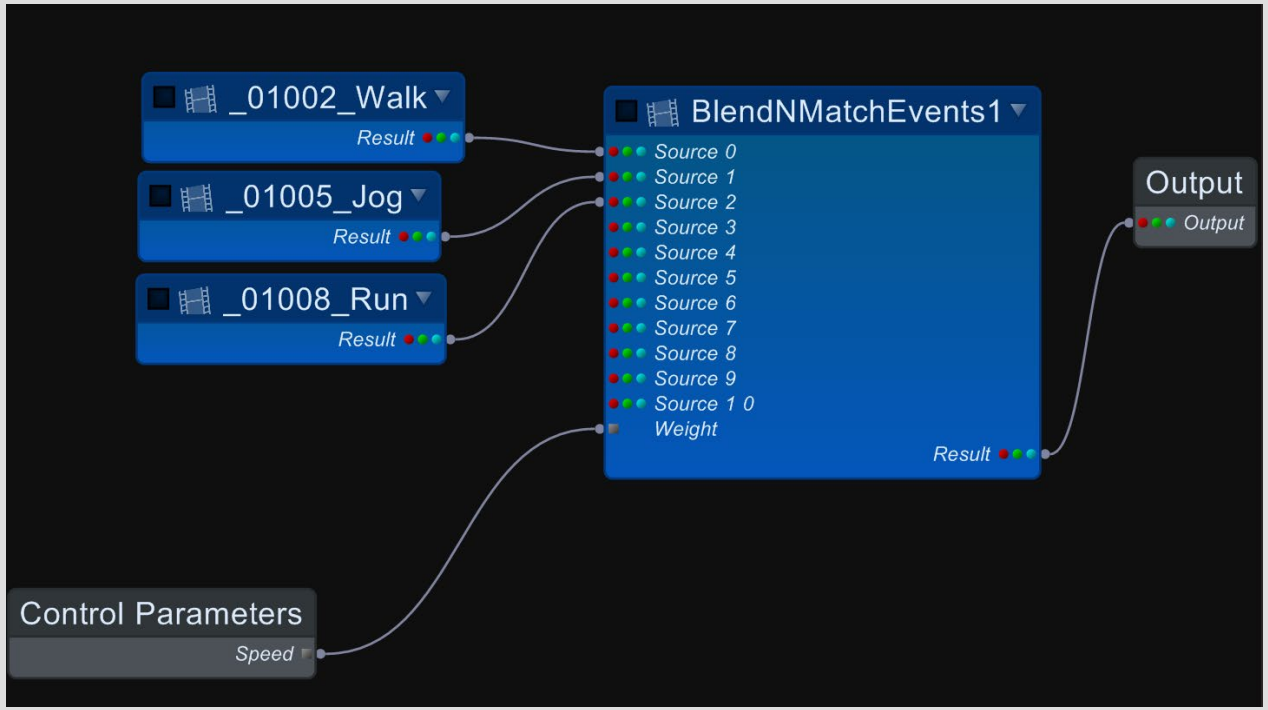
```
stand           {JUMP_PRESS             stand2crouch }
stand2crouch
{
                JUMP_RELEASE            hop
                END                     crouch
}
crouch          {JUMP_RELEASE           takeoff }
takeoff         {END                    float }
hop             {END                    float }
float           {NEAR_GROUND            land }
land            {END                    stand }
```

Creating State Machines

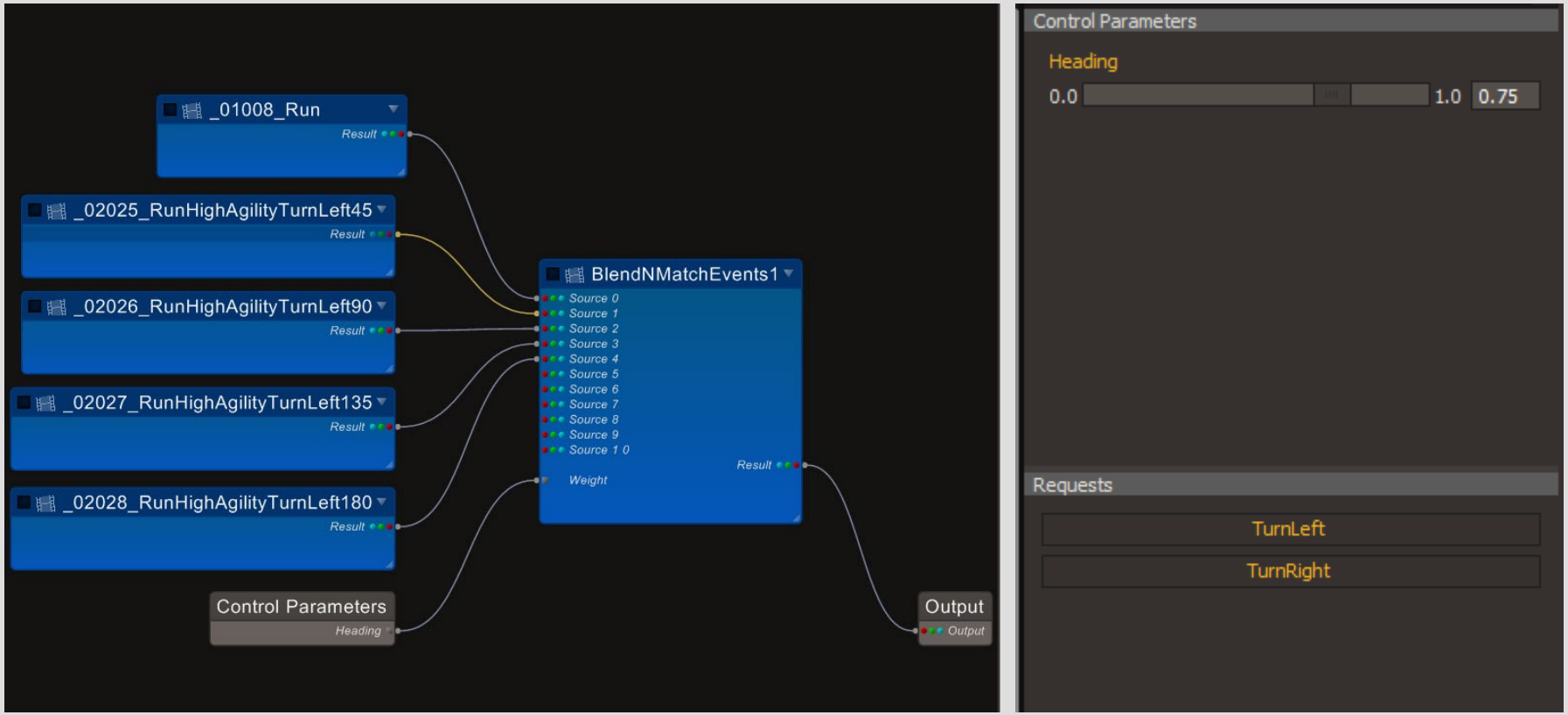
- Typing in text
- Graphical state machine editor
- Automated state machine generation (motion graphing)

State Machine Example

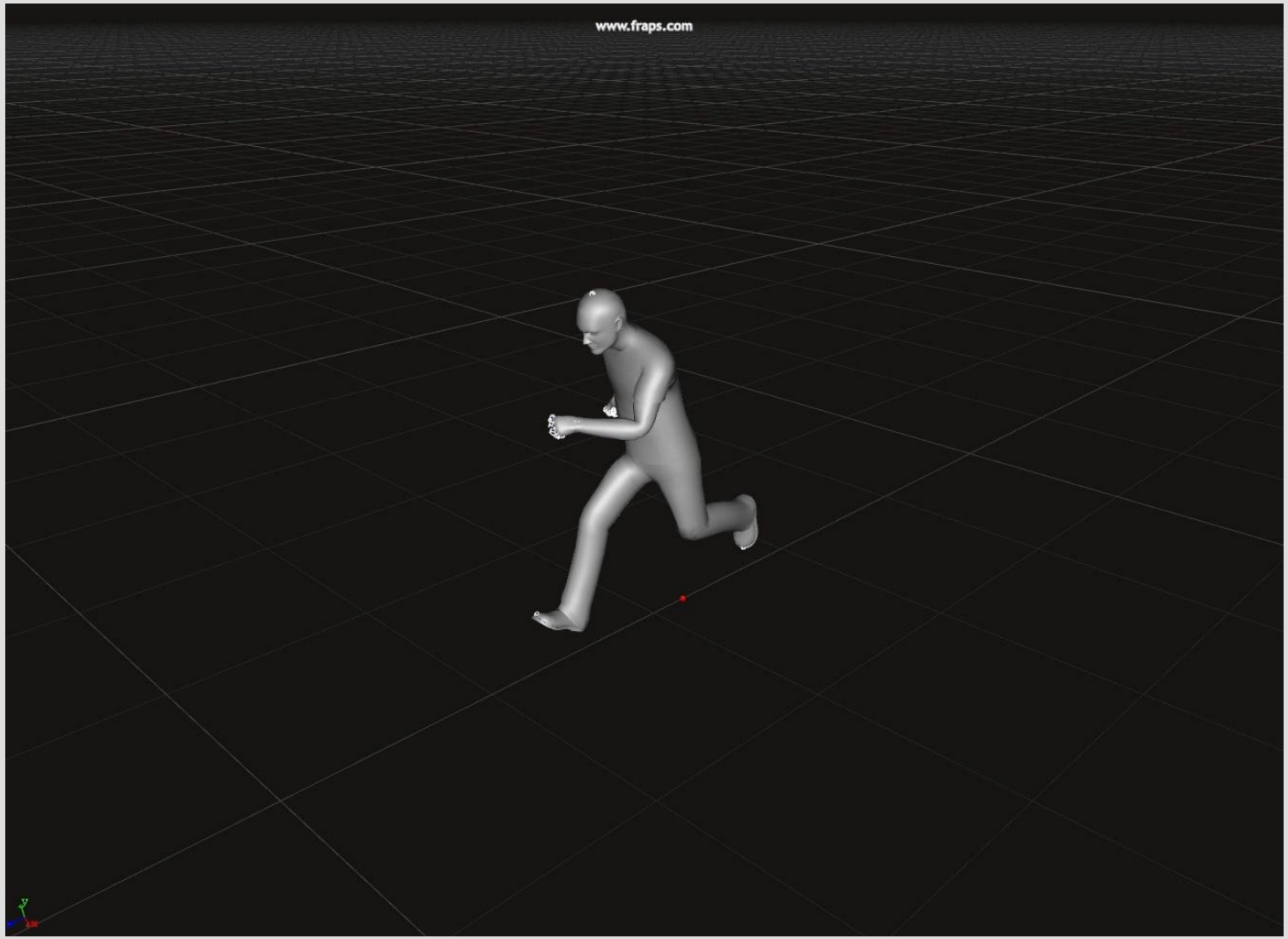
- Motions: Walk, Jog, Run
- Blending parameter: speed
  - 0.0: Walk
  - 0.5: Jog
  - 1.0: Run

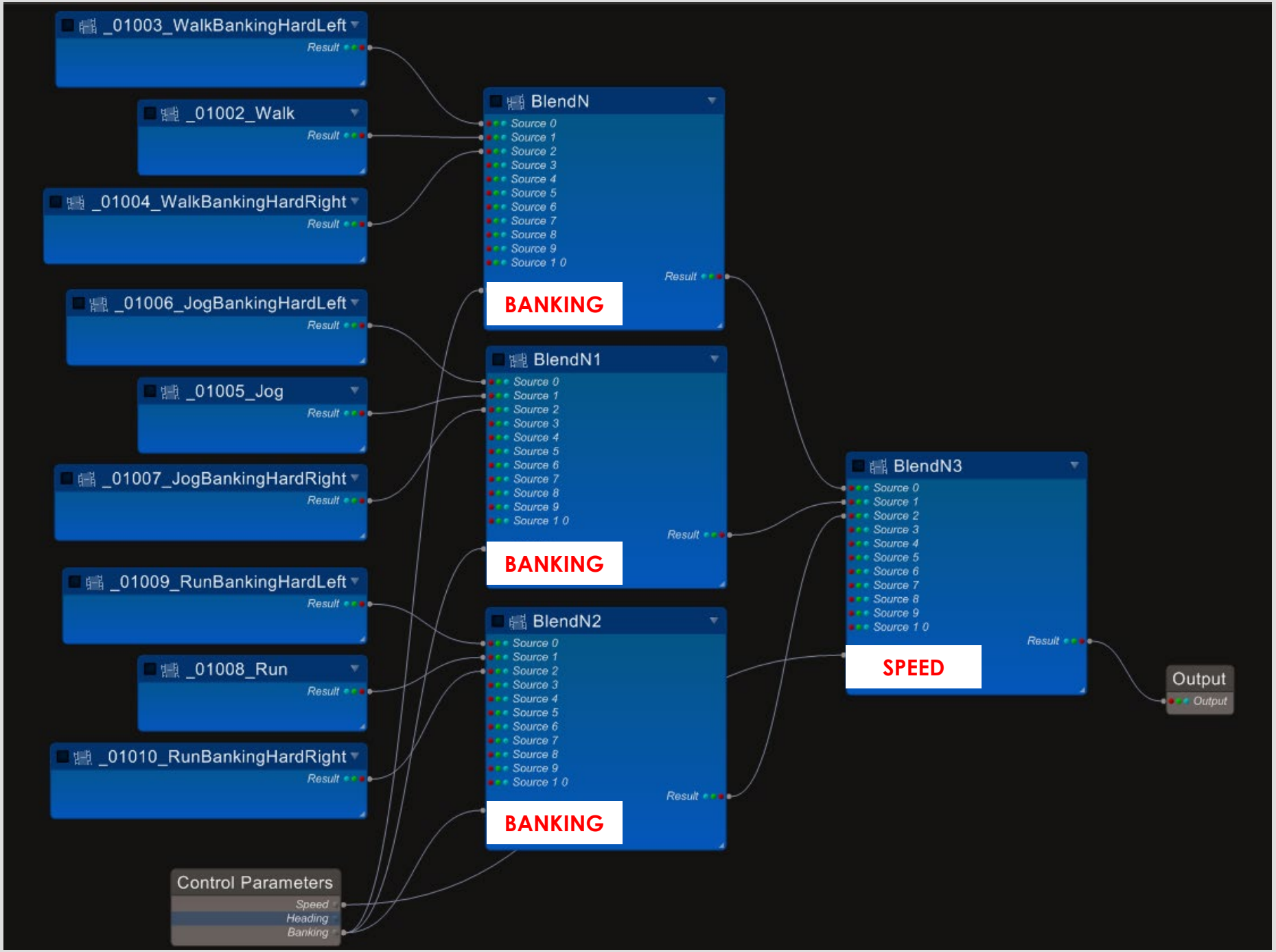


Turning


_01008_Run
Result
_02025_RunHighAgilityTurnLeft45
Result
_02026_RunHighAgilityTurnLeft90
Result
_02027_RunHighAgilityTurnLeft135
Result
_02028_RunHighAgilityTurnLeft180
Result
Control Parameters
Heading
BlendNMatchEvents1
Source 0
Source 1
Source 2
Source 3
Source 4
Source 5
Source 6
Source 7
Source 8
Source 9
Source 1 0
Result
Weight
Output
Output
Control Parameters
Heading
0.0
1.0
0.75
Requests
TurnLeft
TurnRight

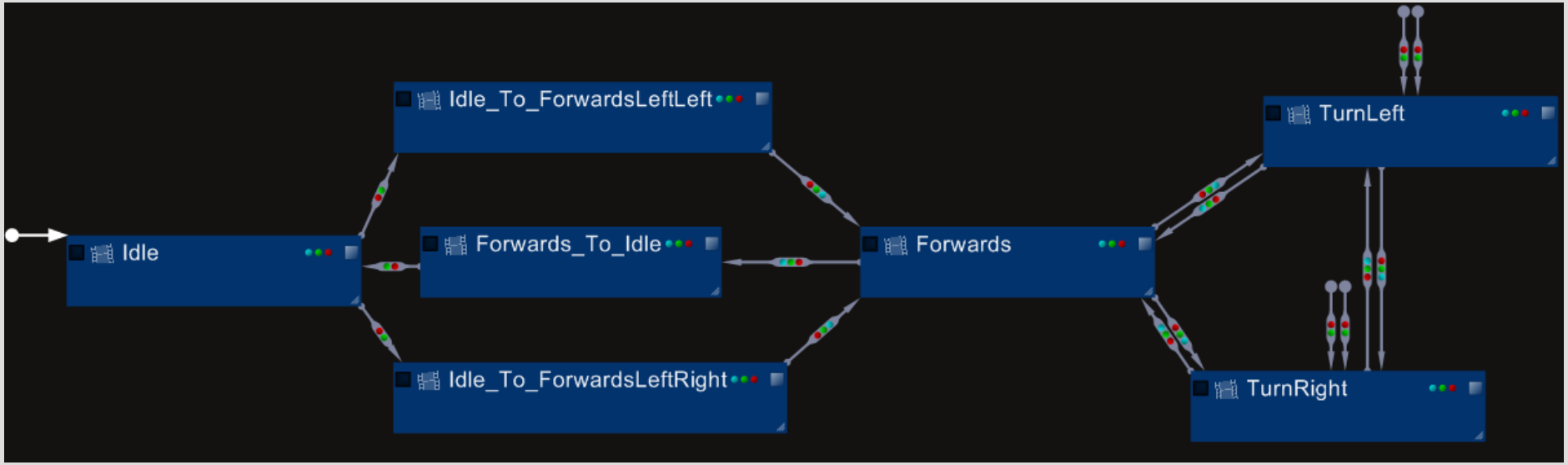
Result




_01003_WalkBankingHardLeft
Result
_01002_Walk
Result
_01004_WalkBankingHardRight
Result
_01006_JogBankingHardLeft
Result
_01005_Jog
Result
_01007_JogBankingHardRight
Result
_01009_RunBankingHardLeft
Result
_01008_Run
Result
_01010_RunBankingHardRight
Result
Control Parameters
Speed
Heading
Banking
BlendN
Source 0
Source 1
Source 2
Source 3
Source 4
Source 5
Source 6
Source 7
Source 8
Source 9
Source 1 0
Result
BANKING
BlendN1
Source 0
Source 1
Source 2
Source 3
Source 4
Source 5
Source 6
Source 7
Source 8
Source 9
Source 1 0
Result
BANKING
BlendN2
Source 0
Source 1
Source 2
Source 3
Source 4
Source 5
Source 6
Source 7
Source 8
Source 9
Source 1 0
Result
BANKING
BlendN3
Source 0
Source 1
Source 2
Source 3
Source 4
Source 5
Source 6
Source 7
Source 8
Source 9
Source 1 0
Result
SPEED
Output
Output

Nested State Machine

- It is also possible to combine State Machines together



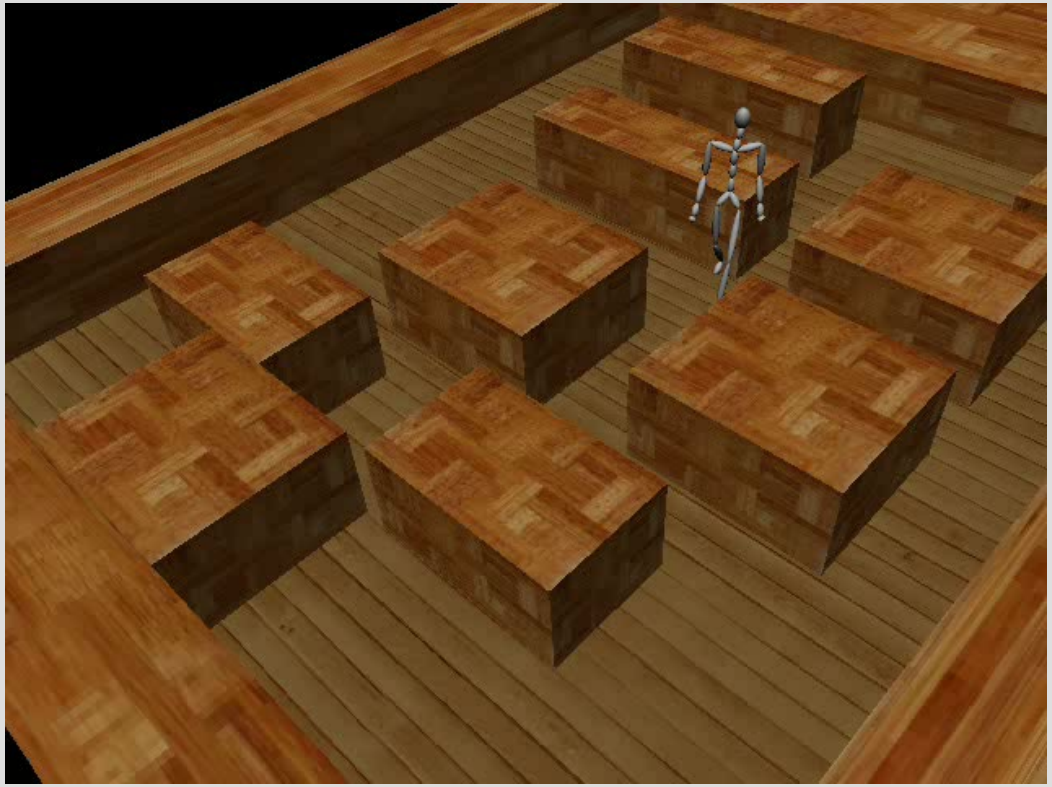
Using Motion Data

- Problems
  - Captured data can be voluminous
  - Processing motion data is intensive
  - Capturing all possible motion is impossible
  - Motion Capture is expensive and cumbersome

- Solutions
  - Organize and represent data
  - Combine data intelligently to synthesize new motion
  - Simulate physics to dynamically generate new motion

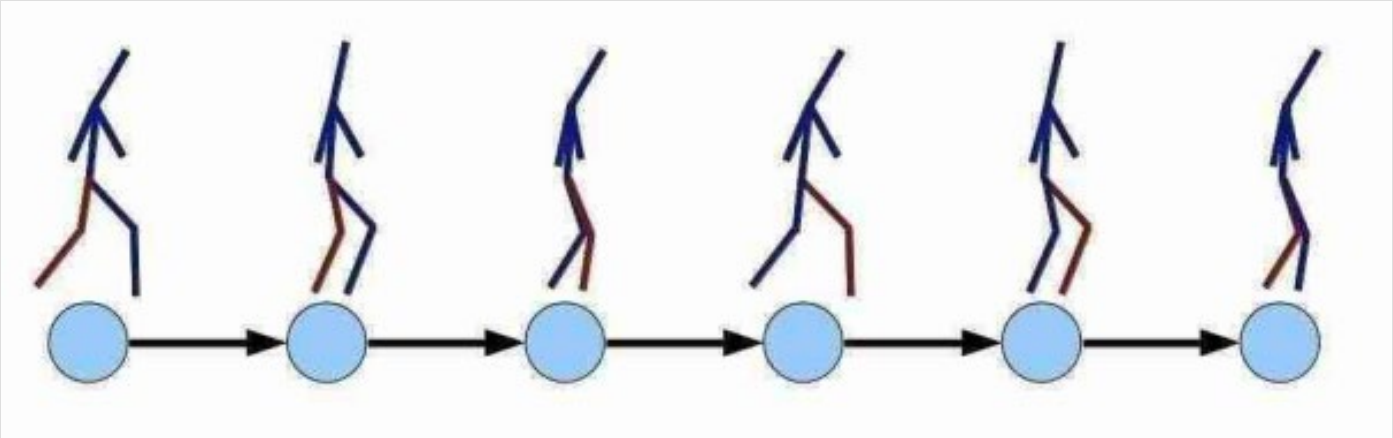
Using Motion Data

- Retain realism of motion capture while giving a user the ability to control and direct a character



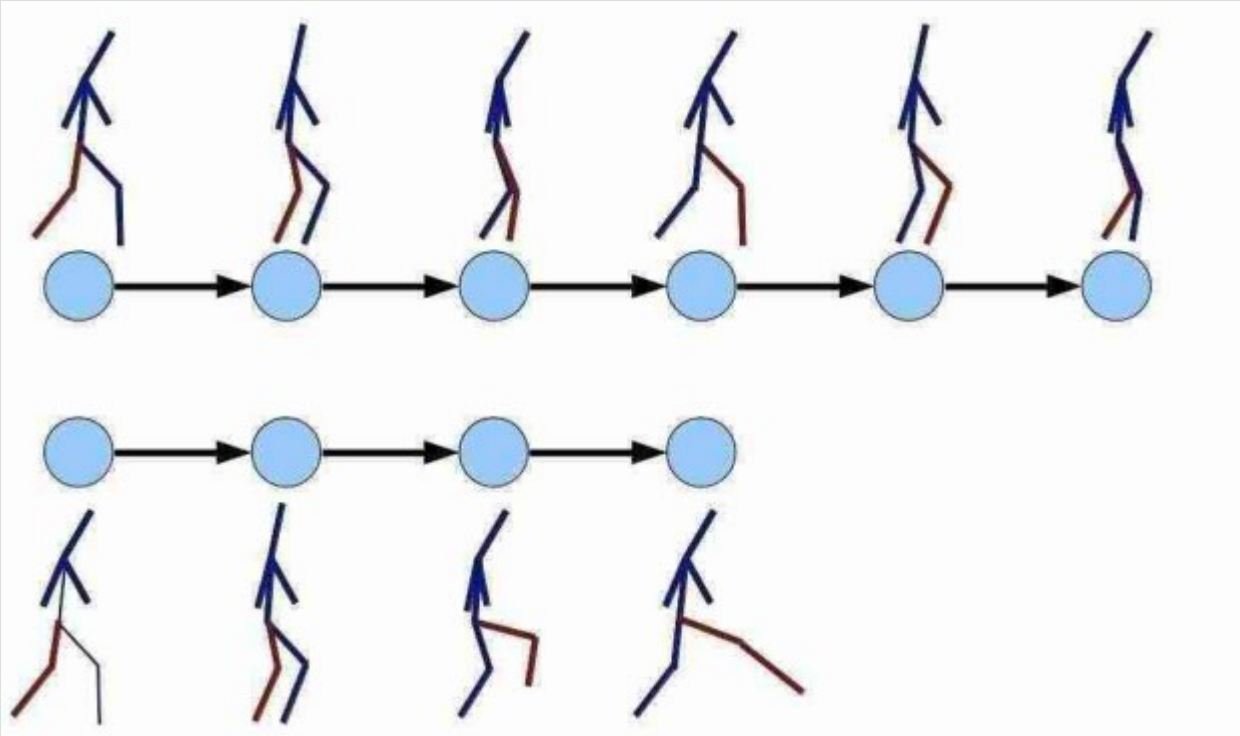
Idea

- Every motion clip is a graph
  - Node: pose
  - Edge: transition frames



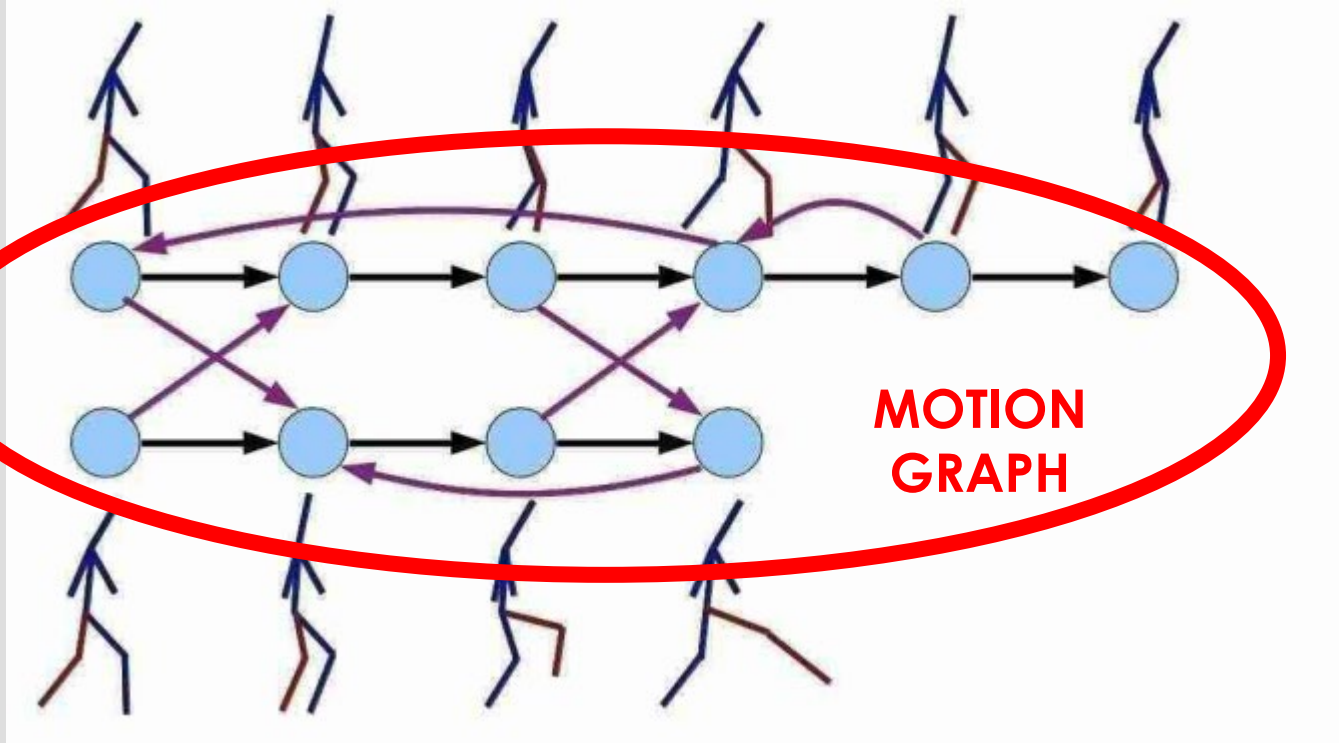
Idea

- There are many such clips in a motion database.



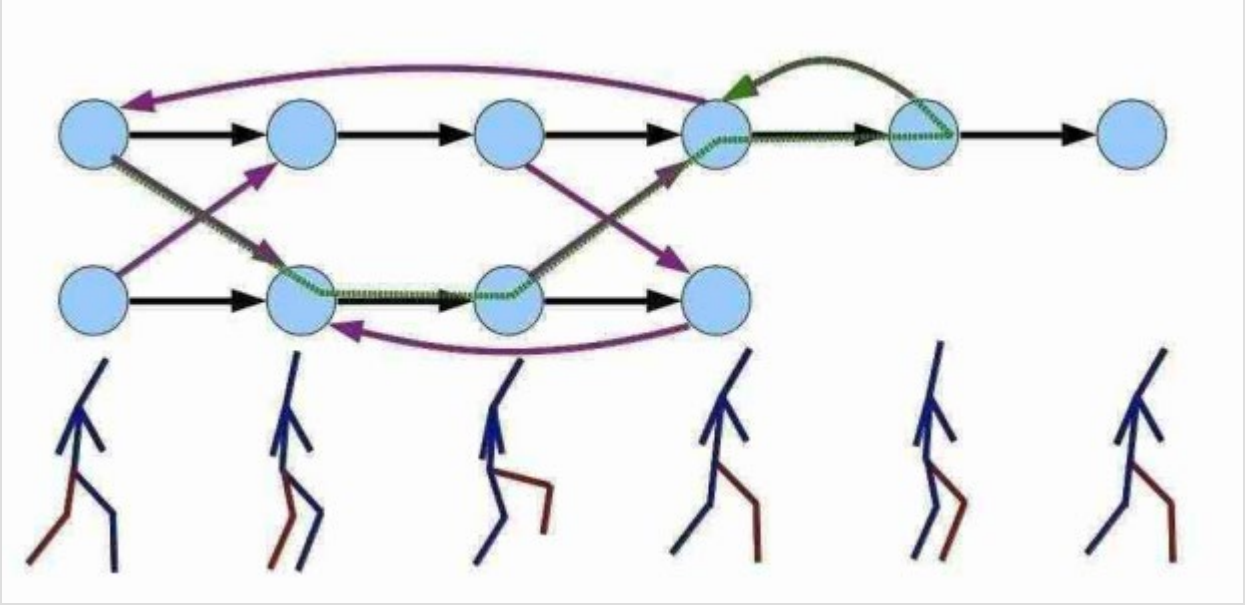
Idea

- Find similar poses between clips
  - Add transitions between them



Idea

- Now any walk on this graph...
  - ...generates a new, smooth motion



Construction

- Similarity between poses across clips
  - Simple distance measure between joints: bad idea
    - Some joints have more influence on the pose
    - Some joints may also be subject to constraints
  - A seamless transition must account not only for differences in body posture
    - but also for differences in joint velocities, accelerations, and possibly higher-order derivatives.

Example Motion Graph

- Source motions:
  - Run:
  - Duck:
  - Punch:

Step 1: Build Similarity Matrices

For each frame i in running motion
For each frame j in ducking motion
cost(i,j) = posecost(i,j)
+ velocitycost(i,j)
+ accelcost(i,j)
similarity(i,j) = 1/(1+cost(i,j)

Cost Functions

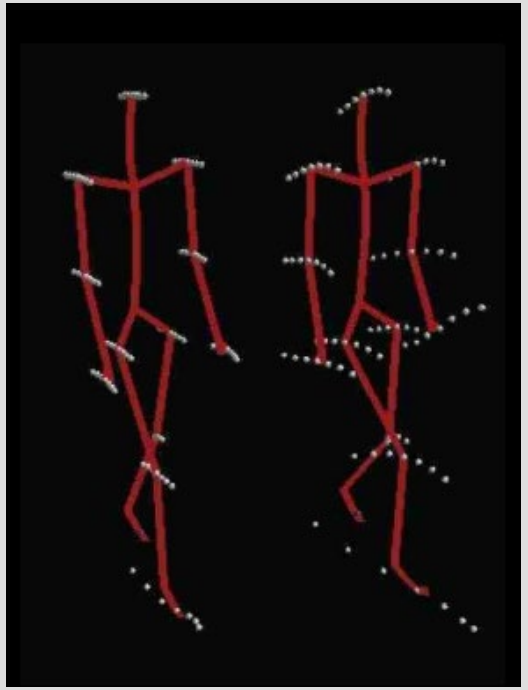
- posecost(i,j) = Σ w(joint)* | joint(i)-joint(j) |
  - w(hip) = 1.0
  - w(shoulder) = 0.75
  - w(knee) = 0.25
  - w(elbow) = 0.25
  - w(else) = 0.0

- i.e., use gross limb movement to distinguish between motion types

Cost Functions

- velocitycost(i,j) = Σ w(vel)*w(joint)* | jointvel(i)-jointvel(j) |
  - jointvel(a) = joint(a)-joint(a-1)
  - w(vel) = 0.1

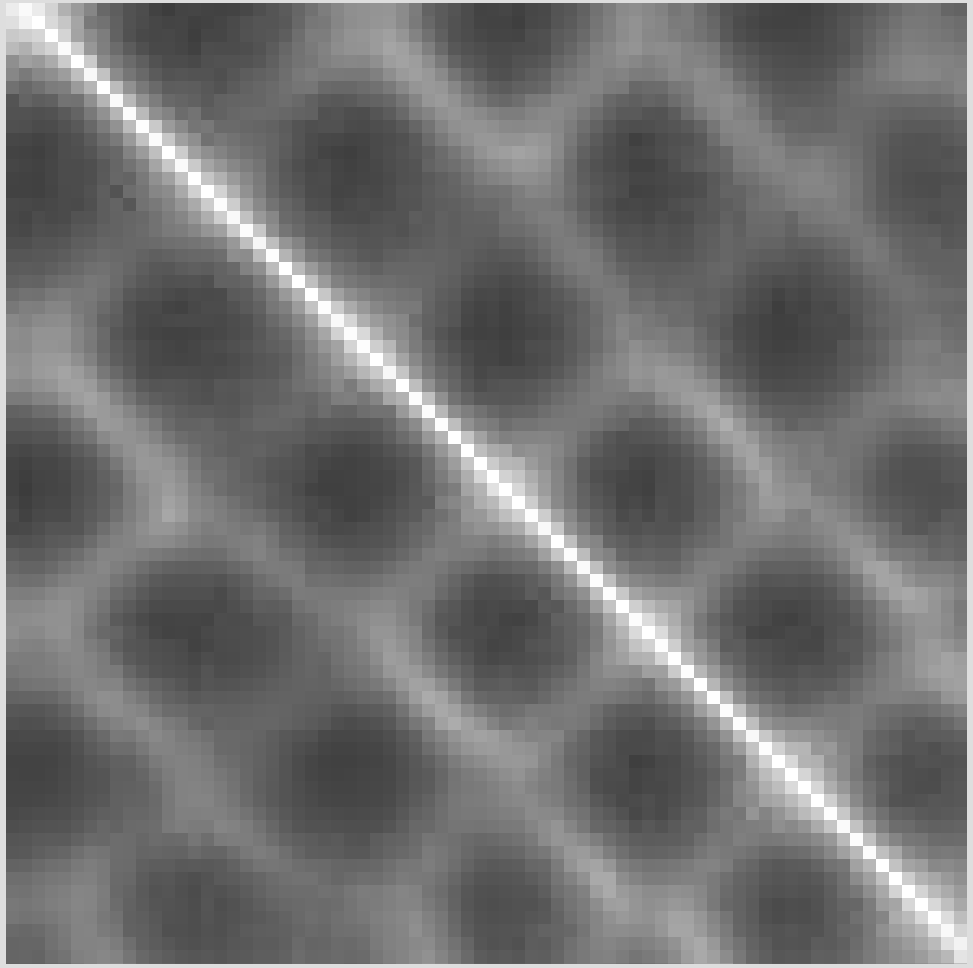
- i.e., take into account the velocities of the key joints, but to a lesser extent than pose



Cost Functions

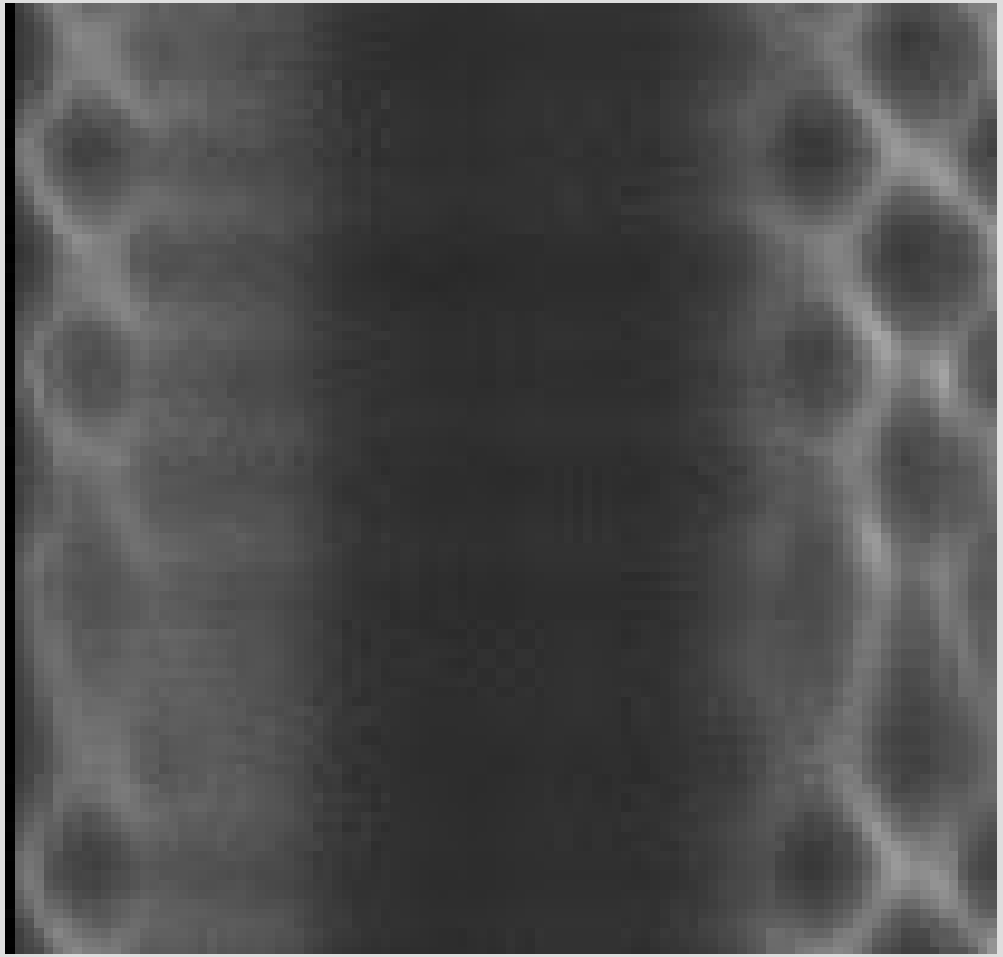
- accelcost(i,j) = $\sum$ w(acc)*w(joint)* | jointacc(i)-jointacc(j) |
  - jointacc(a) = jointvel(a)-jointvel(a-1) = joint(a) – 2*joint(a-1) + joint(a-2)
  - w(acc) = 0.01

Similarity Matrices



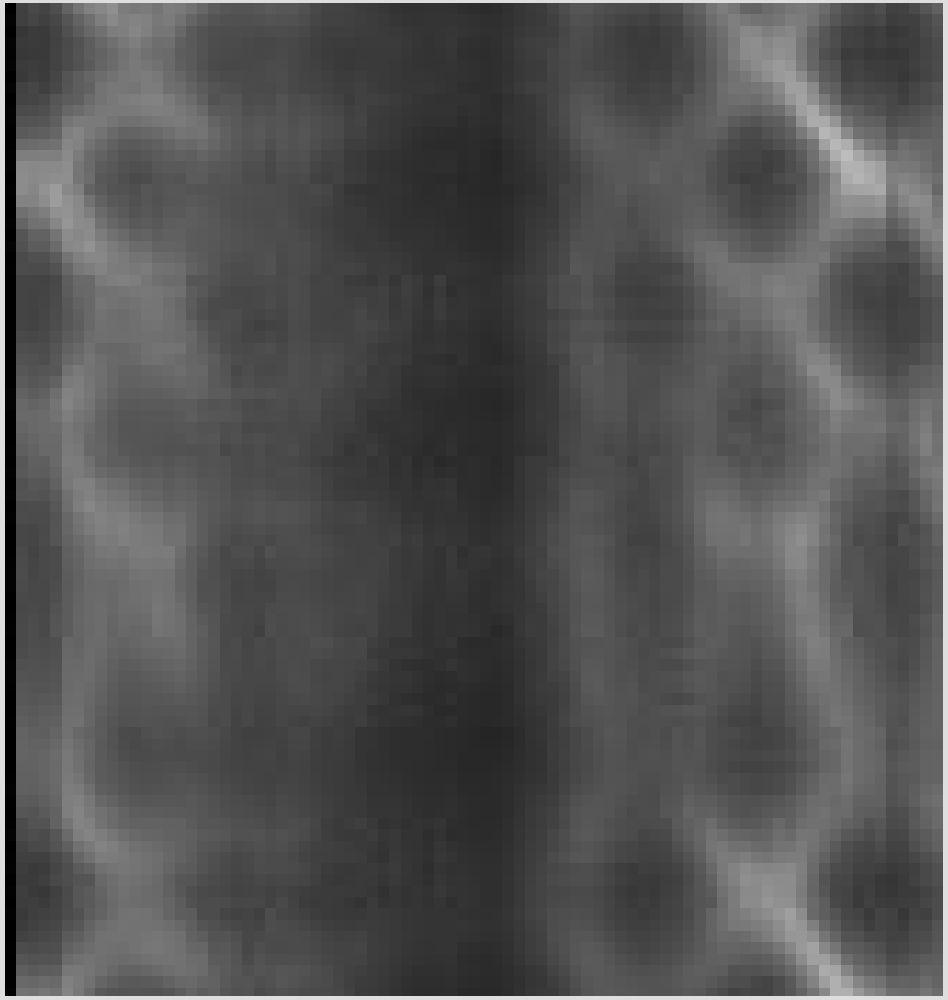
Running vs. Running

Similarity Matrices



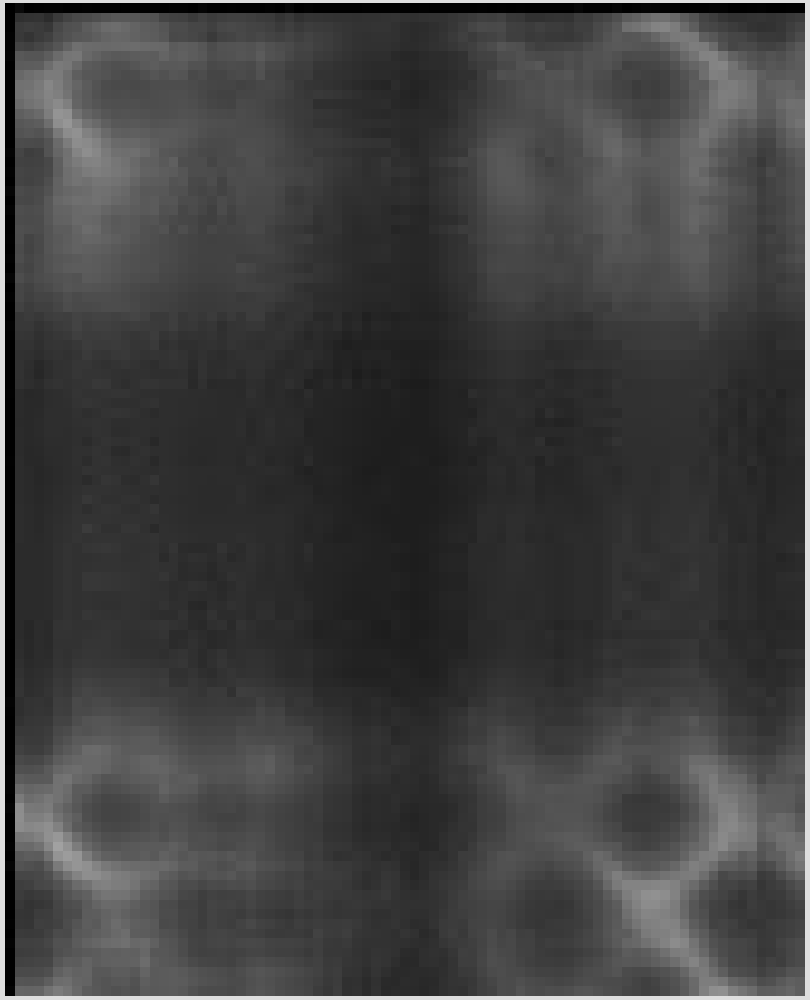
Running vs. Ducking

Similarity Matrices



Running vs. Punching

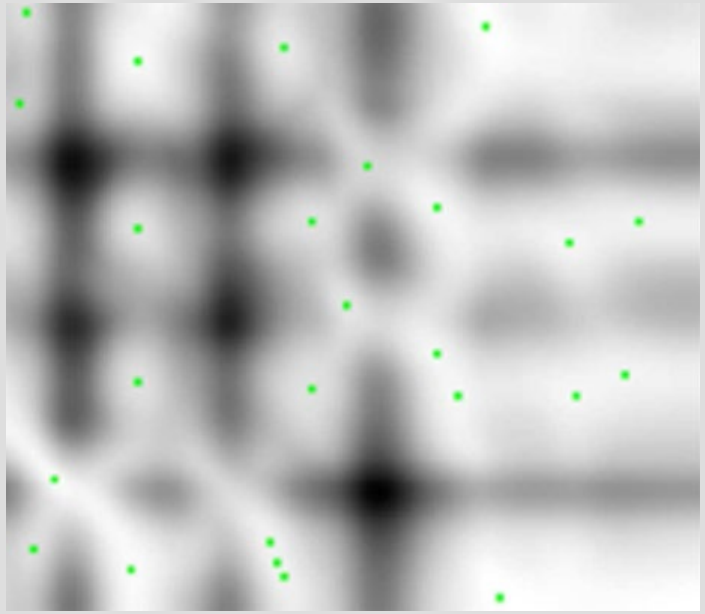
Similarity Matrices



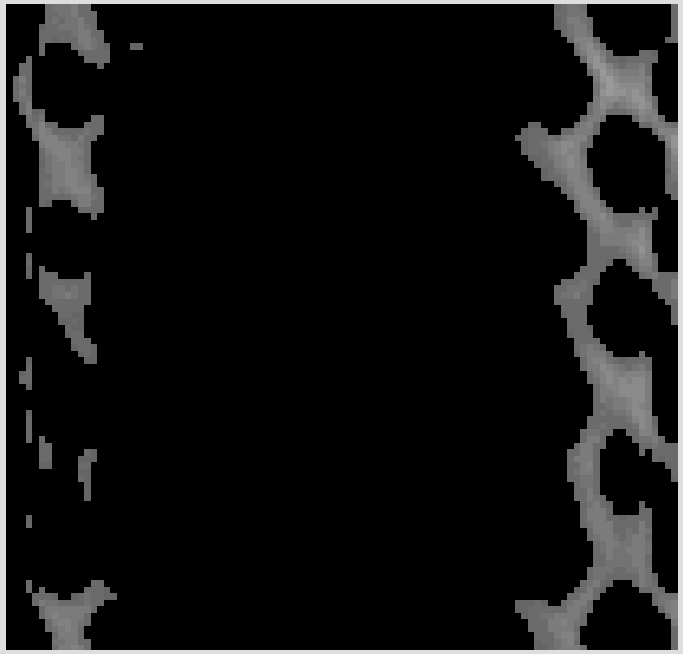
Ducking vs. Punching

Similarity Matrices

- High-quality motions tend to cluster
  - Near-transitivity
- Want only a *representative* of each cluster
  - Remove values which are not local maxima



Thresholded Matrices



Threshold 0.7

Local Maxima

Running vs. Ducking

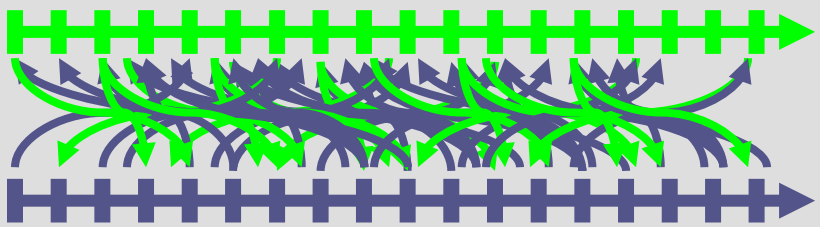
Basic Motion Graph

1. Acquire motion
2. Create similarity matrices
3. Cull matrices to local maxima, threshold
4. Build graph
5. De-seam transitions at runtime

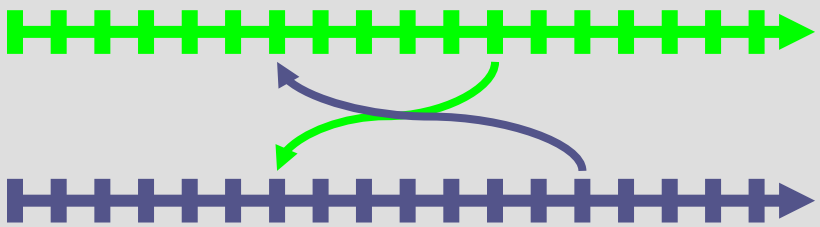
Motion Graph Benefits

- Re-use of high-quality input
  - Minimal editing, so maintain characteristics
- Automatic
  - Large amounts of data
  - Novice users
  - Natural transitions between behaviours
  - As-good-as-possible added transitions
- Fast and easy to use at runtime

Standard Motion Graph Creation



Low threshold:
too many transitions



High threshold:
too few transitions

Motion Graphs

- Problem: hard to get a specific result
  - Optimizable
  - Or just tweak it. A lot.

Searching for a motion

- Random walks on the motion graph are not interesting
  - So we search for motion that satisfies some objective
- Minimize a function f (w) such that

$$f(w) = f([e_1, \cdots e_n]) = \sum_{i=1}^{n} g([e_1, \cdots e_{i-1}], e_i)$$

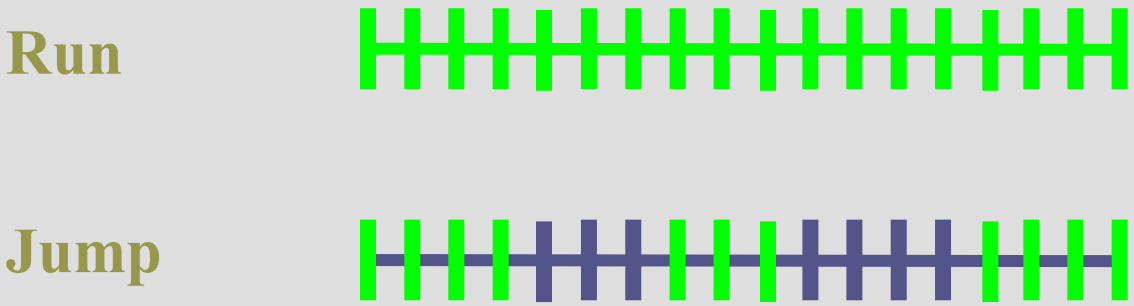
- Goal: find a complete graph walk *w* that minimizes *f*
  - *g* should give some guidance throughout the entire motion, as arbitrary motion is not desirable

Path Synthesis

- Project root onto the floor at each frame, forming a piecewise linear curve
- Minimise the distance between optimal and actual path created by sequence of motion clips
- Confine the search to the subgraph containing appropriately labelled motion

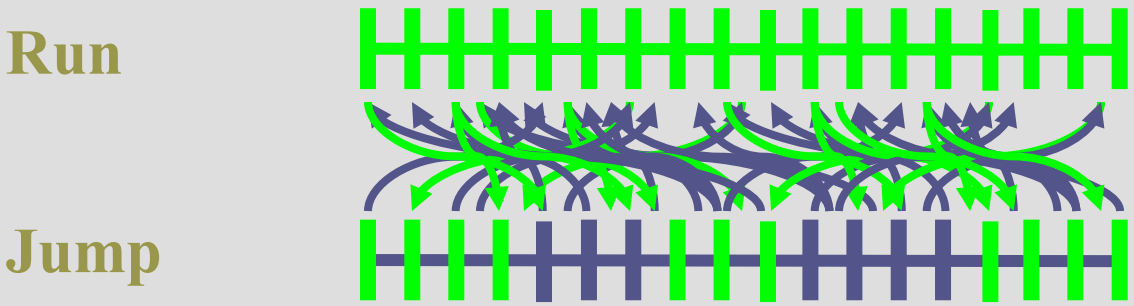
Other Issues

- Suppose you have a motion graph...



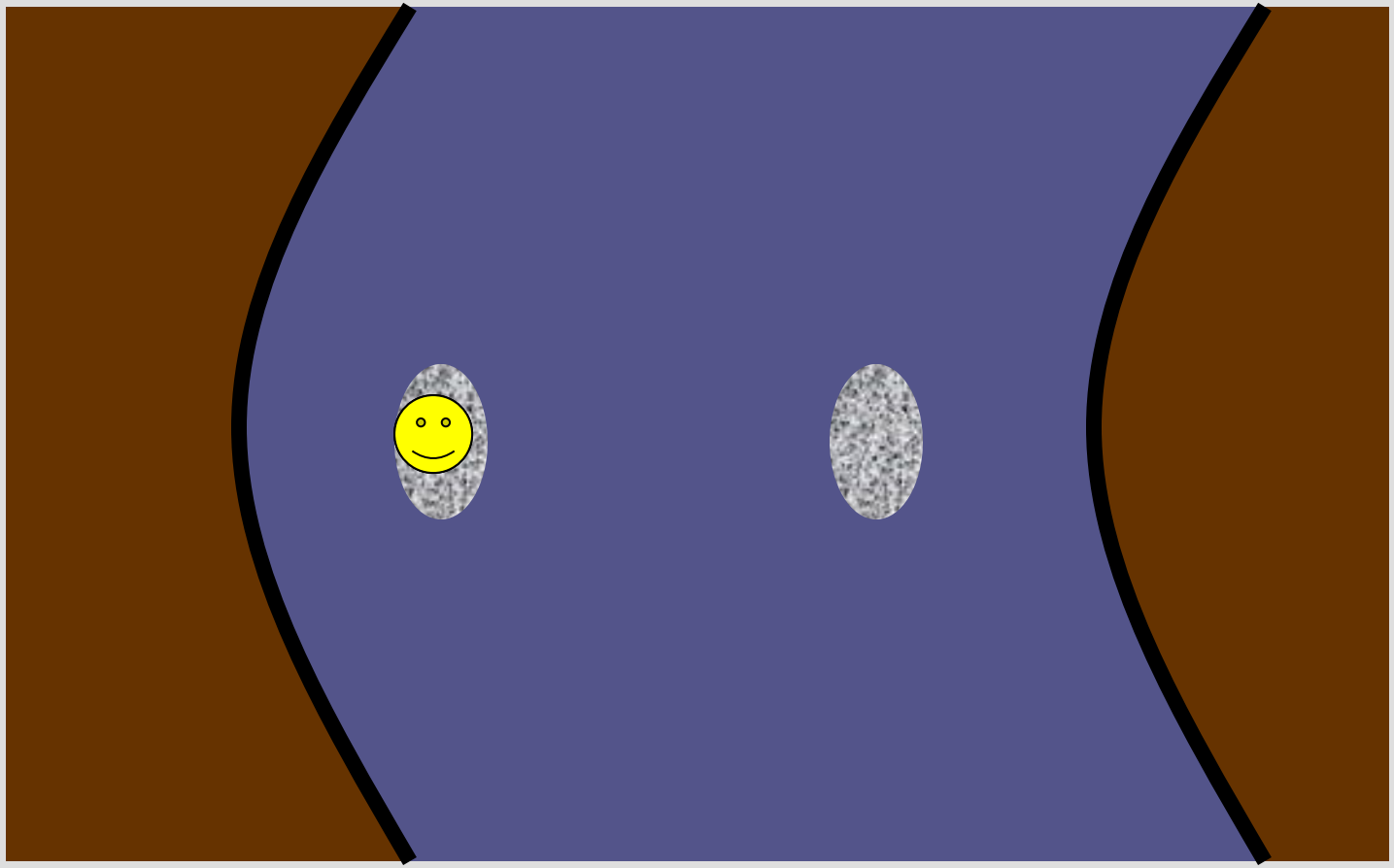
Jumping Puzzle

- Suppose you have a motion graph…

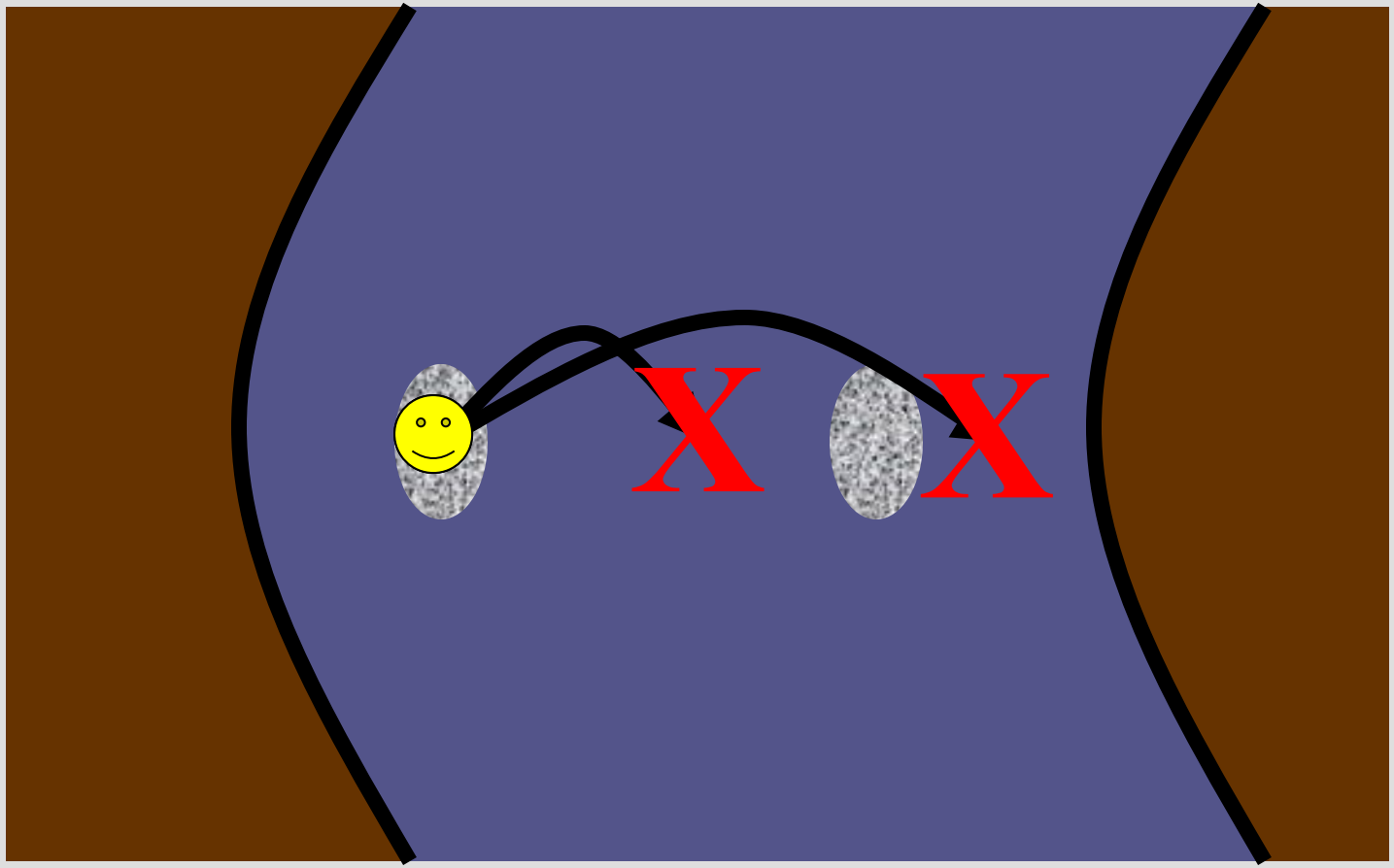


- …with plenty of transitions.

Jumping Puzzle

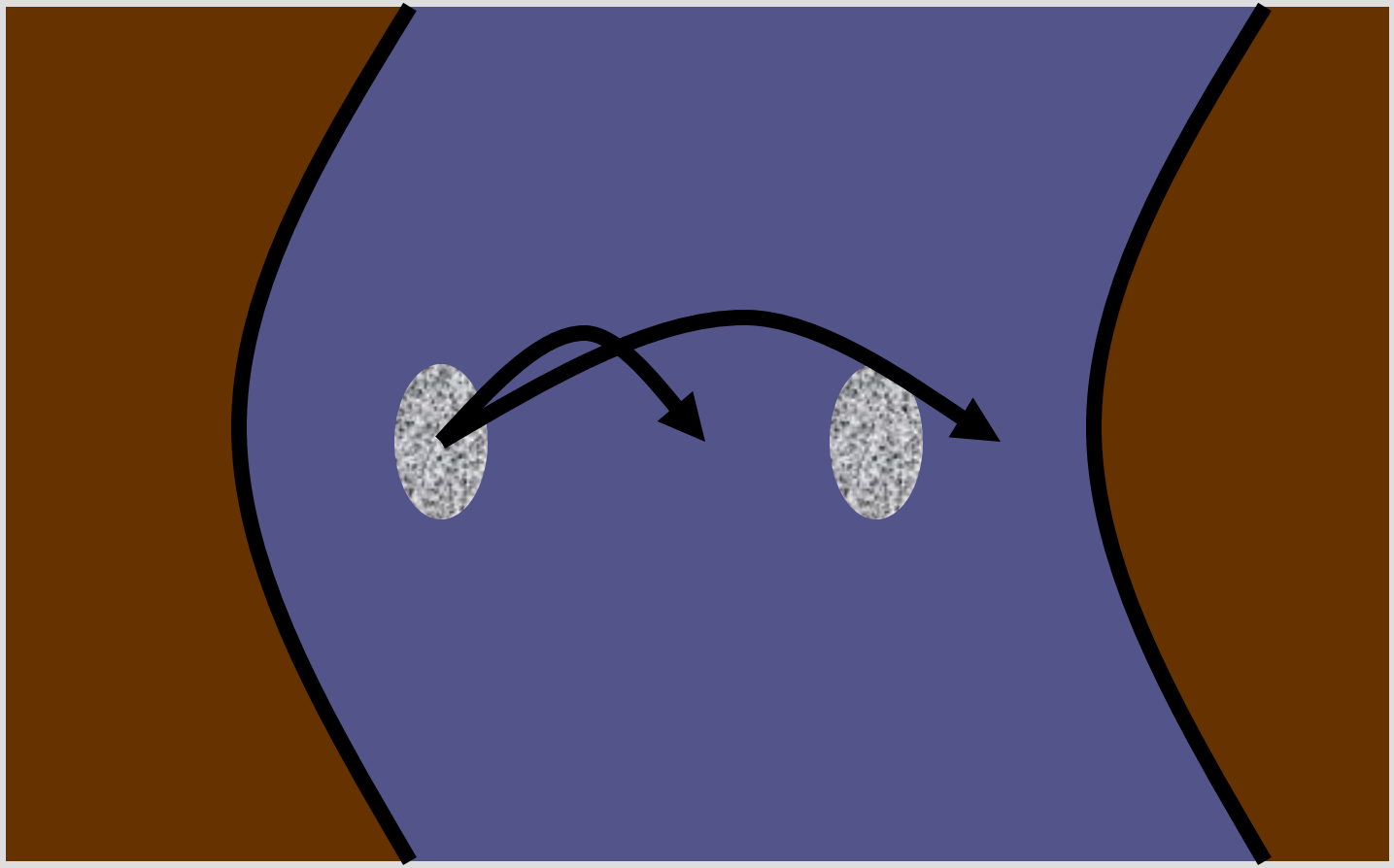


Jumping Puzzle



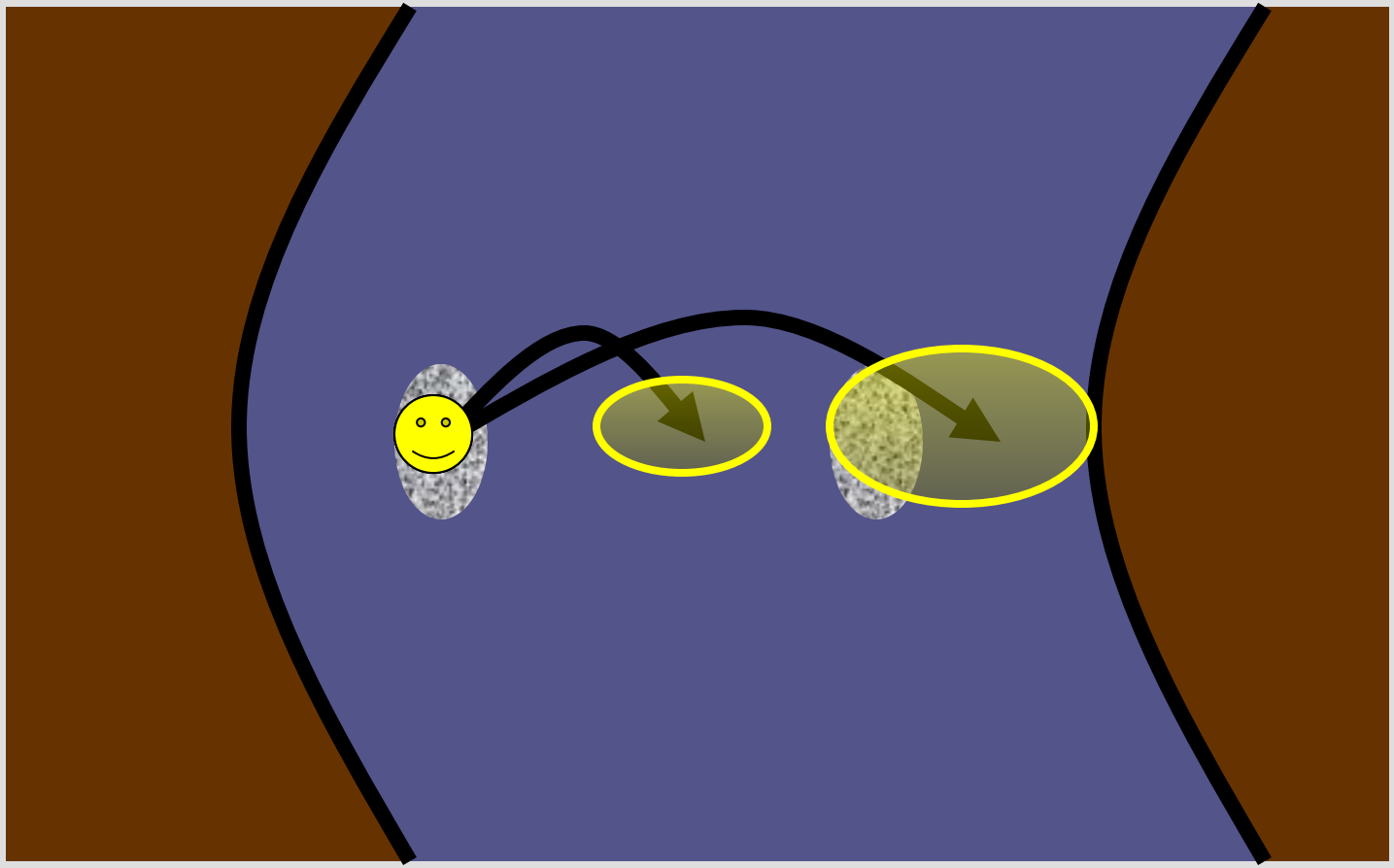
Available source motions can’t perform task

Motion Graph Problems



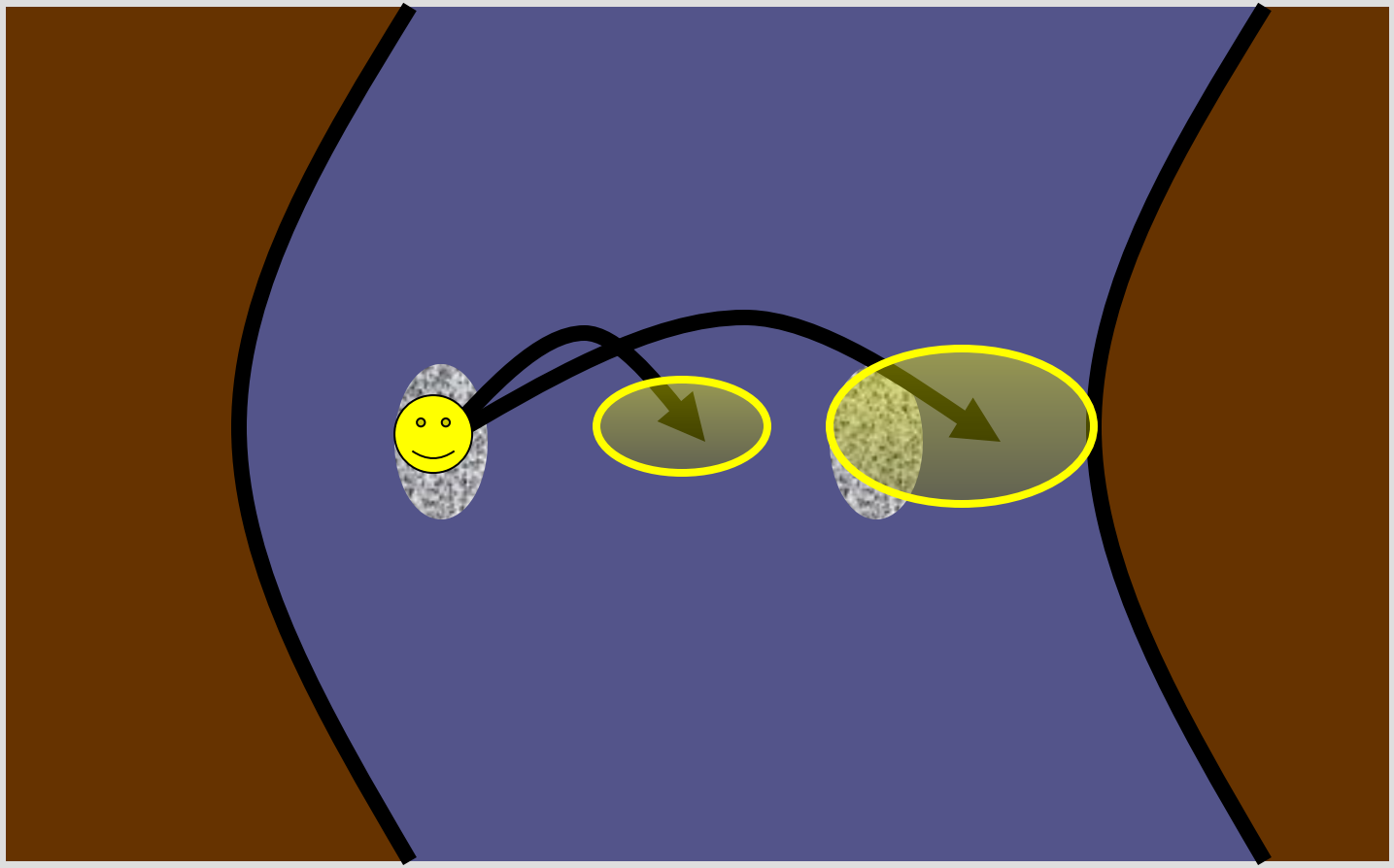
Lack of **motion capability**

Motion **Warping**



Warp the motion to change jump's landing point

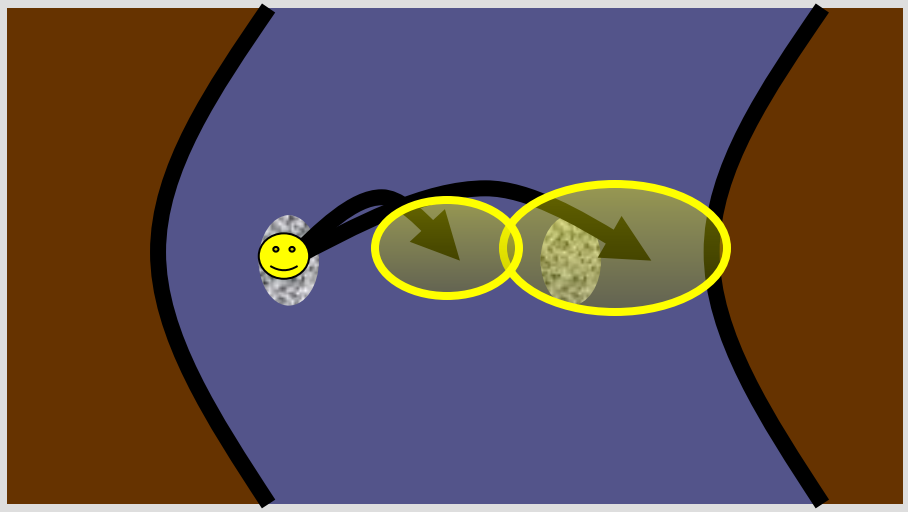
Motion Quality



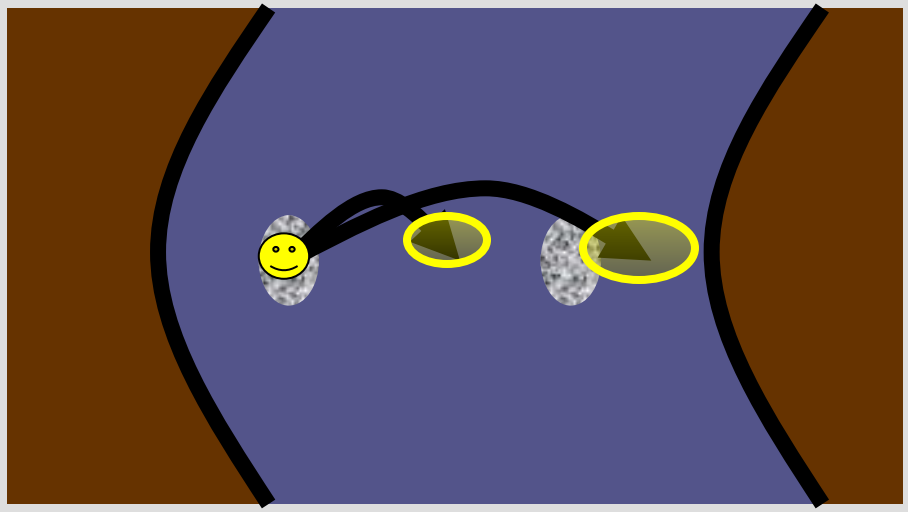
Too much motion editing degrades quality

Capability vs. Quality Tradeoff

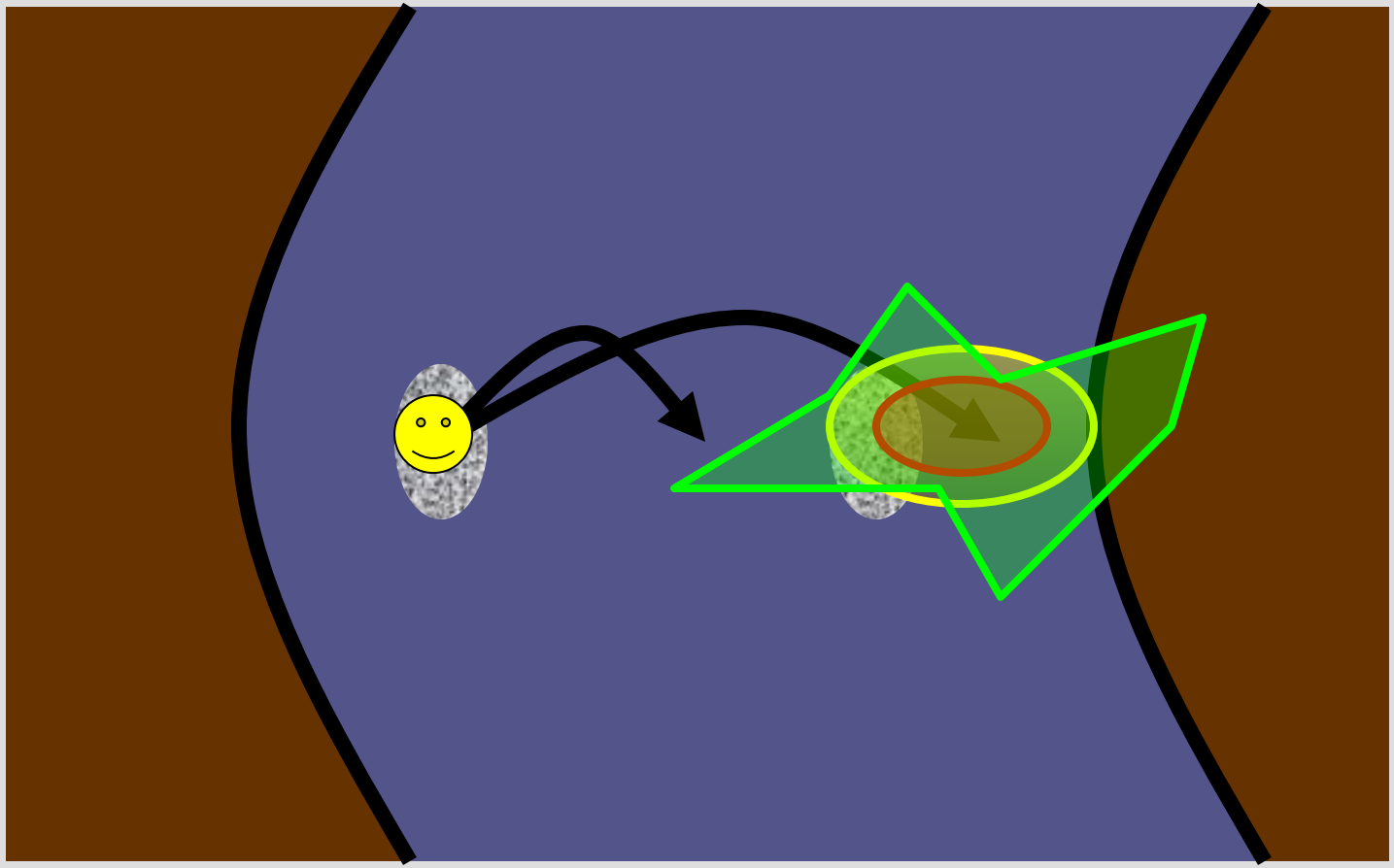
- Low motion quality okay



- High motion quality required



What is quality motion?



How much editing is acceptable?

Motion Quality

- Another aesthetic problem
  - Cannot be computed from the motion data alone
  - Collect data from people, fit a model

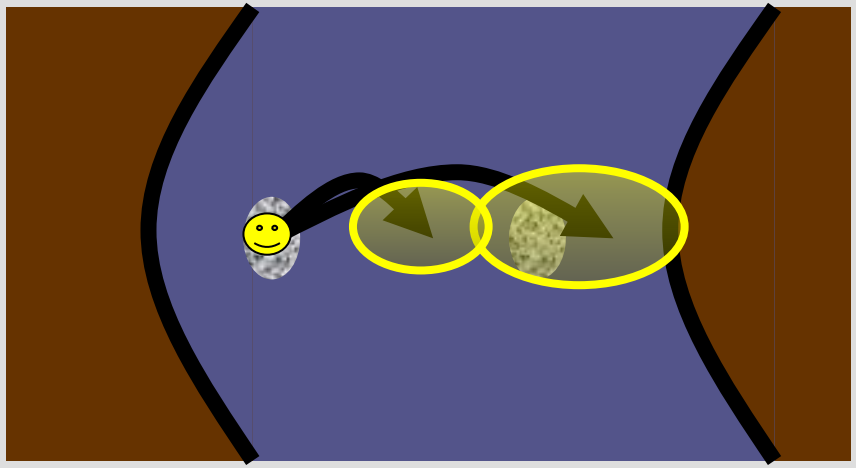
Motion Quality

- Another aesthetic problem
  - Cannot be computed from the motion data alone
  - Collect data from people, fit a model

- A model of human perception of motion?
  - Tall order, new effort
  - Guided by success in other areas (rendering)

Approach: Measure Perceived Error

- How noticeable are these errors?
  - How do we measure perceived error?
- How much error is acceptable?



Meaning?

- Where does that leave the character?
- Pretty but dumb
  - Basic motion graphs give great motion but poor control
- Controllable but at risk
  - Blend trees risk degrading the animation even away from transitions
  - Current best bet for highly interactive

Conclusions

- Motion Graphs are very useful in character animation
- Extensively used for real-time animation synthesis
- Newer methods include a combination of physics simulation and motion capture
- Latest research: Neural State Machines (next week)

More Information

- Motion Graphs
  - "Motion Graphs", Kovar and Gleicher, 2002
  - "Evaluating Data-Driven…", Reitsma and Pollard, 2007
- Retargetting
  - https://www.youtube.com/watch?v=Vn-vVzMGgec
  - "Retargetting Motion to New Characters", Gleicher 1998
- Perception
  - "Perceptual Metrics…", Reitsma and Pollard, 2003
  - "Evaluating the Visual…", O'Sullivan et al. 2003
  - "…Quantifying Natural Human Motion", Ren et al., 2005
- Blend Trees
  - www.gamasutra.com/features/20030704/edsall_03.shtml