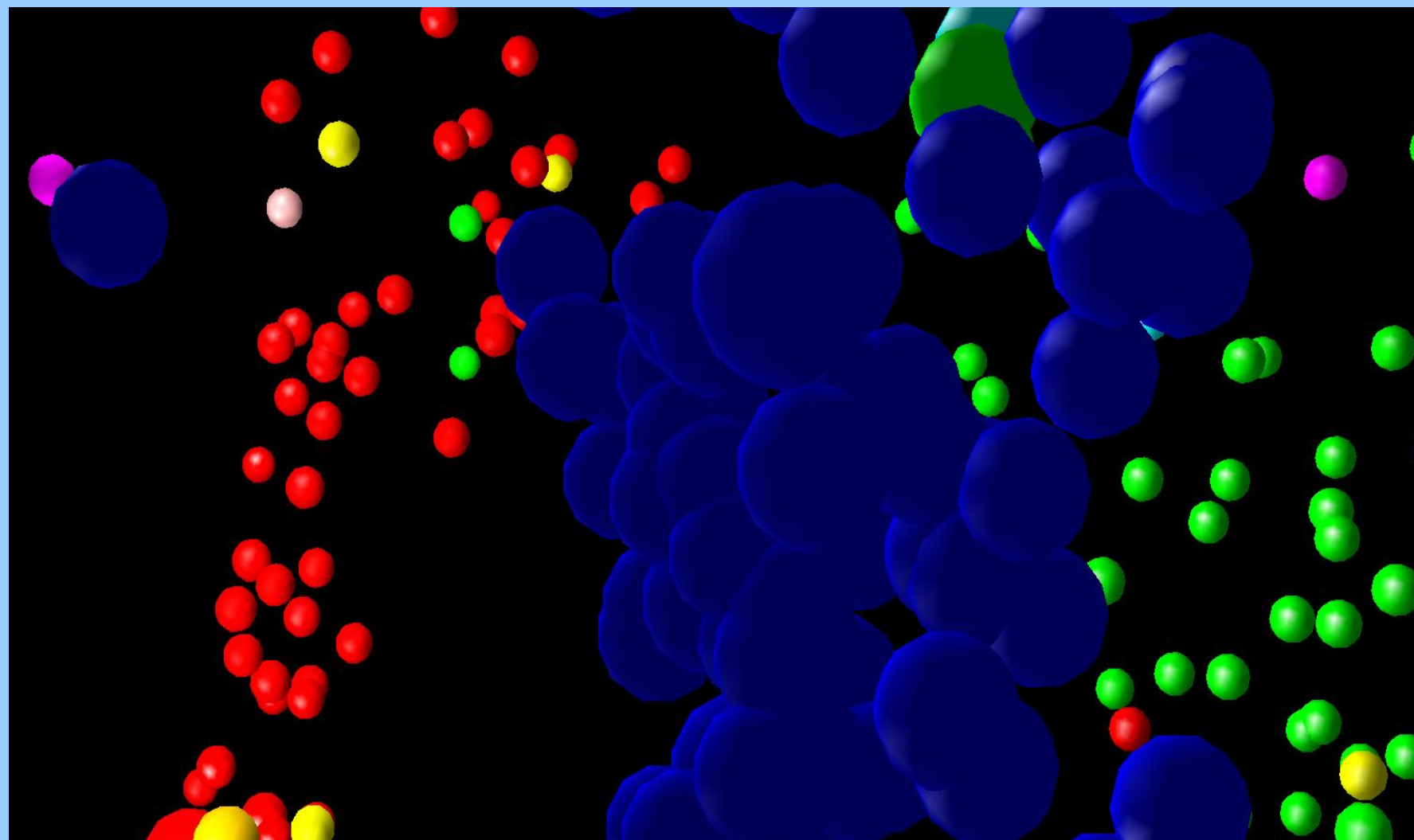


# Using 3D Visualizations to Explore and Discover Music

Paul Lamere, Douglas Eck  
Sun Microsystems

**Search Inside the Music** is an application for exploring and discovering new music. **Search Inside the Music** uses a music similarity model and 3D visualizations to provide a user with new tools for exploring and interacting with a music collection. With **Search Inside the Music**, a music listener can find new music, generate interesting playlists, and interact with their music collection.

## 3D Visualization of the Music Space



- Written in the Java Programming Language
- Uses the Java3D API
- Songs positioned in 3D space using Multidimensional scaling

### Multidimensional scaling

```
randomlyPlaceTracksInPlottedSpace();

while (totalStress > STRESS_THRESHOLD) {
  for (int i = 0; i < tracks.length - 1; i++) {
    for (int j = i + 1; j < tracks.length; j++) {
      Track t1 = tracks[i];
      Track t2 = tracks[j];

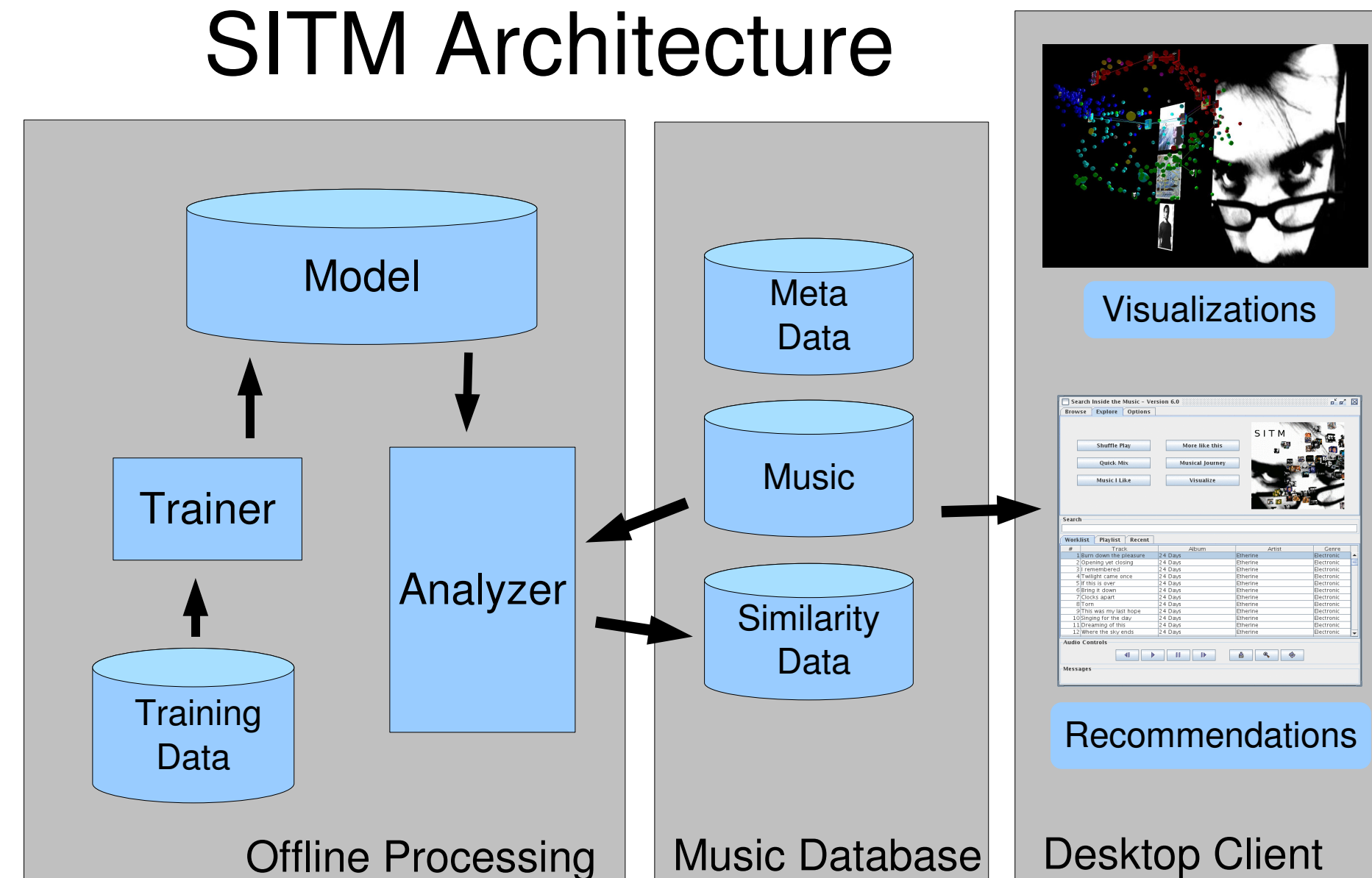
      genreDist = getGenreSpaceDistance(t1, t2);
      plotDist = getPlottedDistance(t1, t2);
      delta = genreDist - plotDist;
      force = delta * k;
      totalStress += Math.abs(force);
      pushApart(t1, t2, force);
    }
  }
  // This is expensive! O(n^3)
}
```

### MDS Chalmers algorithm

```
while (totalStress > STRESS_THRESHOLD) {
  for (Track t1 : tracks) {
    t1.clearRandomList();
    while (t1.needsRandomTracks()) {
      Track t2 = selectRandomTrack();

      if (t1.isNeighbor(t2)) {
        t1.addToNeighborList(t2);
      } else {
        t1.addToRandomList(t2);
      }
    }
    totalStress += t1.applyForces(neighborList);
    totalStress += t1.applyForces(randomList);
  }
  // Less expensive O(n^2)
}
```

## SITM Architecture

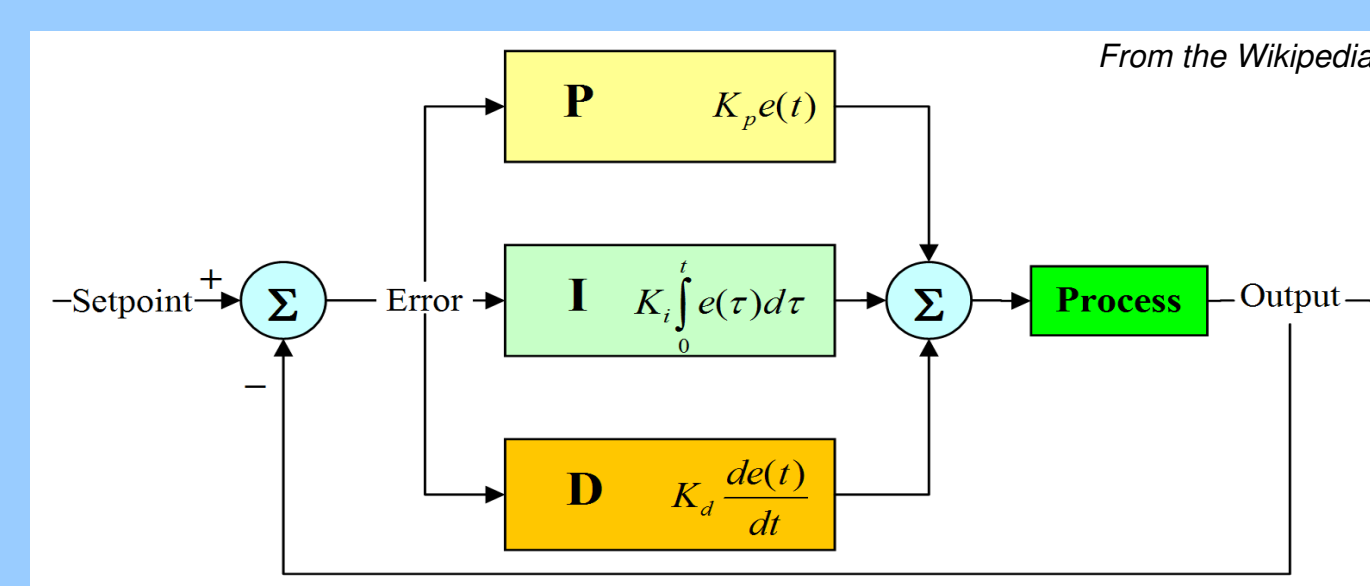


## Flying album art



### Physics model + PID Controller

$$f=ma \quad a=f/m \quad v=v+a \quad s=s+v$$



```
void update() {
  for (int i = 0; i < setPoint.length; i++) {
    double error = setPoint[i] - curPoint[i];
    double delta = lastError[i] - error;
    integrator[i] += error;
    integrator[i] = clamp(integrator[i], -maxInt, maxInt);

    double cmd = kP * error + kI * integrator[i] - kD * delta;

    cmd = clamp(cmd, -maxCmd, maxCmd);
    velocity[i] += cmd;
    velocity[i] = clamp(velocity[i], -maxVel, maxVel);
    curPoint[i] += velocity[i];
    lastError[i] = error;
  }
}
```

## Building the album grid



**Goal:** Arrange the albums so that similar sounding albums are adjacent by minimizing *overall stress*.

- **Overall stress** – the average *album stress* for all albums.
- **Album Stress** – the average acoustic distance of an album to its neighbors.

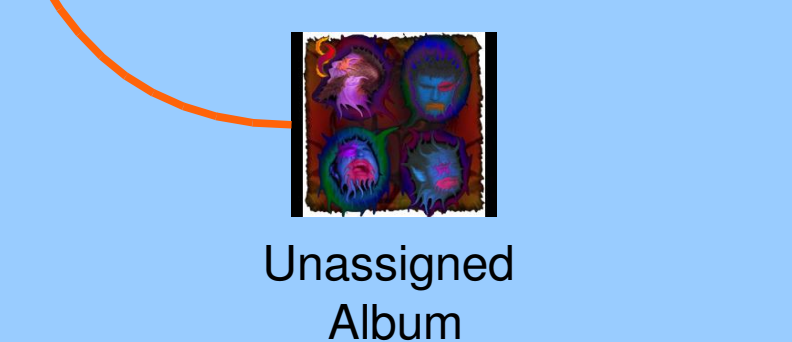
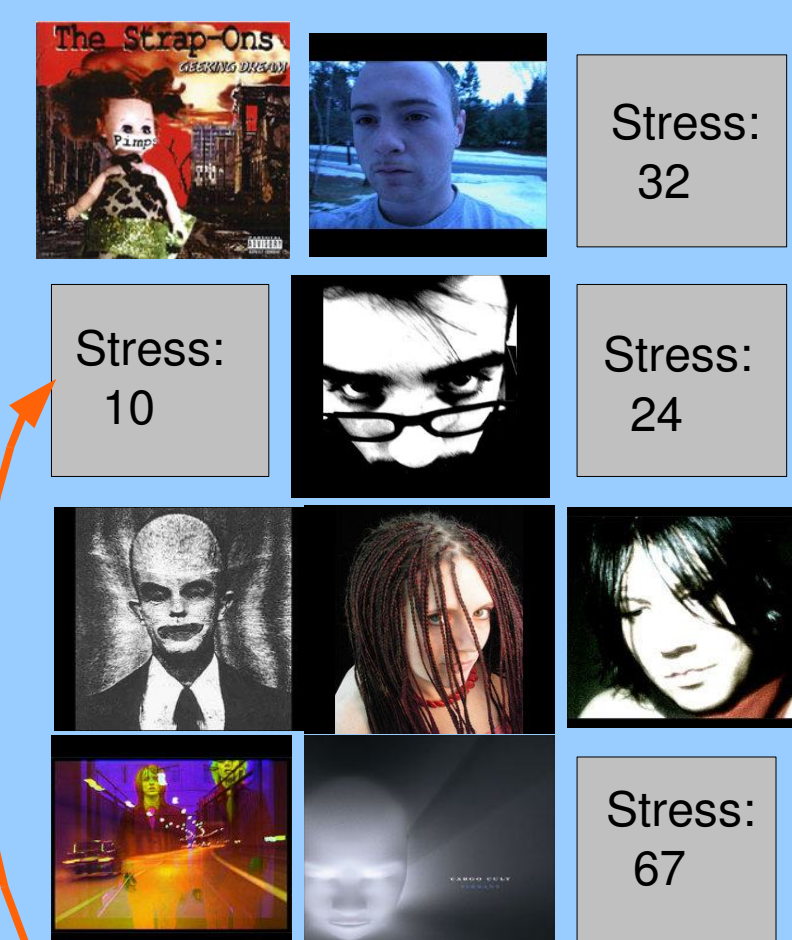
### Four Algorithms

**Random:** Assign all albums to random grid positions.

**Greedy:** Assign albums to the grid sequentially by selecting an album that, when added, will minimize the increase in overall stress. Assign the selected album to the connected grid position that minimizes overall stress.

**Random Swap:** Initial layout with the Random algorithm. Repeatedly select two random albums and swap the positions of the two albums if the swap reduces the overall stress. Stop when the overall stress converges.

**Greedy+RSwap:** Initial layout with the Greedy algorithm followed by Random Swap to further reduce stress.



Algorithm	Overall Stress
Random	1.00
Greedy	0.31
Random Swap	0.22
Greedy+RSwap	0.13

**Using 3D Visualizations to Explore and Discover Music**

Presented at ISMIR 2007

by Paul Lamere and Douglas Eck of Sun Microsystems Laboratories,  
contact: Paul.Lamere@sun.com