

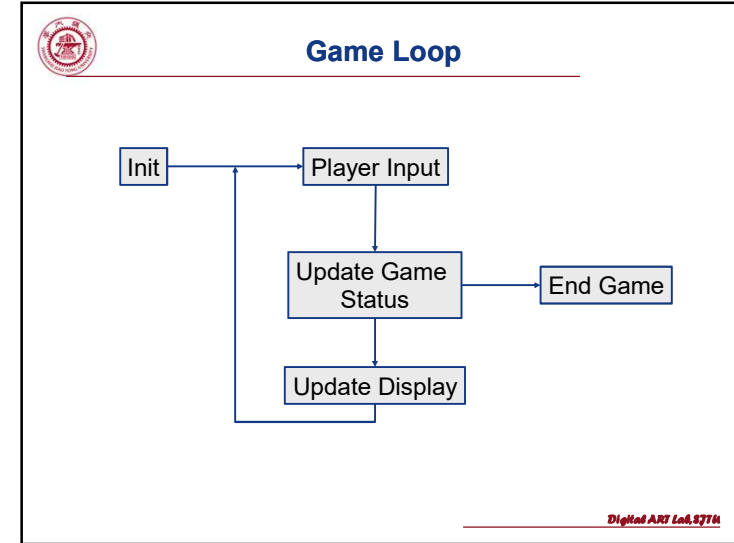
上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

1896 1935 1957 2006

游戏设计与开发 Game Design and Development

2D Game

软件学院数字艺术实验室
Digital ART Lab, SJTU



How Fast Does my Game Loop Need to Run?

ANSWER: "It depends..."

- Visual displays typically need at least 15 Hz for interactivity (25-30 Hz is better)
- Head-tracking for HMDs is OK at 60 Hz, but even only 2-5ms of latency yields *display lag*, which often quickly causes users to lose their lunch...
- Haptic displays usually require much higher update rates (500 -1000 Hz)
- Multitasking / Multiprocessing allows for different update rates for different types of output displays

Digital ART Lab, SJTU

二维游戏技术

- ⊗ 二维游戏概览
- ⊗ 地图的创建与显示
- ⊗ 视差卷轴 (Parallax Scrolling)
- ⊗ 过程化生成
- ⊗ 颜色轮换 (Color Cycling)
- ⊗ 精灵动画 (Sprite Animation)
- ⊗ 碰撞检测

Digital ART Lab, SJTU



二维游戏

- 早期的游戏都是二维的
 - 如Diablo(暗黑破坏神)
 - 只有两个轴(上下, 左右)



- 很多RPG游戏是固定视角的二维半



Diablo ART LoA, 877H



地图的创建与显示

- 为实现一个基本的二维游戏框架, 首先要实现游戏地图的各种加载和编辑操作, 为角色提供游戏环境。
- 4种通用地图实现的方法:
 - 固定地图
 - 滚屏地图
 - 菱形地图
 - 多层次地图

Diablo ART LoA, 877H



固定地图

- 使用固定的背景作为地图
- 将地图切割成棋盘状的一系列小块(Tiles)
- 在内存中保持一个二维数组, 保存每个小块对应的编号
- 绘制时根据数组提供的信息, 在每个小块画上相应图块

1	3	3	4	1	1	2
1	0	0	2	2	0	1
1	0	0	2	2	0	0
1	0	0	0	3	0	3
1	0	0	2	2	0	2
0	0	0	0	0	0	2
3	3	3	3	1	2	2

地图数据

图片(及其编号)

0	1	2	3	4
---	---	---	---	---	-------

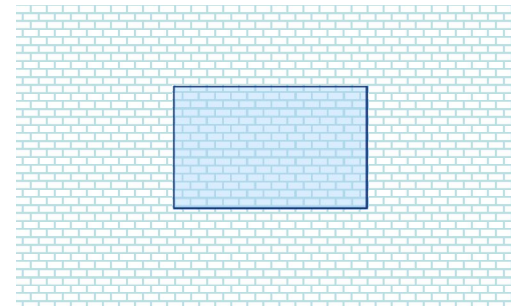
地图的拼接

Diablo ART LoA, 877H



滚屏地图

- 地图大小超出屏幕范围
- 每一时刻只显示一部分地图
- 通过水平移动或垂直方向移动地图, 实现滚屏效果

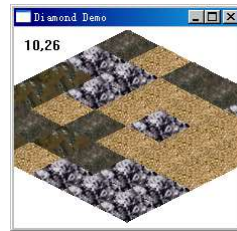
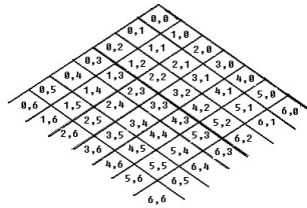


Diablo ART LoA, 877H



菱形地图

- 菱形地图是在二维画面上表现三维场景的常用技术(伪3D, 2.5D)



- 拼接所使用的小地图是菱形，计算比较复杂

Digital ART Lab, SZTH



Games (RTS/TBS/RPG)



Diablo 2



Age of Empires

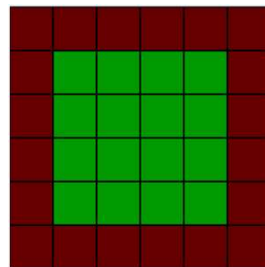
Digital ART Lab, SZTH



Tile-based Games



```
[[1,1,1,1,1,1],
 [1,0,0,0,0,1],
 [1,0,0,0,0,1],
 [1,0,0,0,0,1],
 [1,0,0,0,0,1],
 [1,1,1,1,1,1]]
```



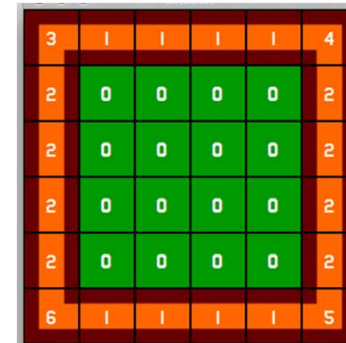
Digital ART Lab, SZTH



More different tiles

```
[[3,1,1,1,1,4],
 [2,0,0,0,0,2],
 [2,0,0,0,0,2],
 [2,0,0,0,0,2],
 [2,0,0,0,0,2],
 [6,1,1,1,1,5]]
```

```
for (i, loop through rows)
  for (j, loop through columns)
    x = j * tile width
    y = i * tile height
    tileType = levelData[i][j]
    placetile(tileType, x, y)
```



Digital ART Lab, SZTH

Types of Projections

④ 透视投影 vs. 平行投影

正投影 vs. 轴测投影

正投影

轴测投影

Digital ART Lab, SZTH

轴侧投影(axonometric projection)

④ 等轴测投影(Isometric Projection)

④ 正二侧投影(Dimetric)

④ 正三侧投影(trimetric)

Digital ART Lab, SZTH

"Isometric" Game

90 degrees

45 degrees

30 degrees

0 degrees

26.5650 degrees

Digital ART Lab, SZTH

Cartesian-Isometric mapping

```

1 //Cartesian to isometric:
2
3 isoX = cartX - cartY;
4 isoY = (cartX + cartY) / 2;

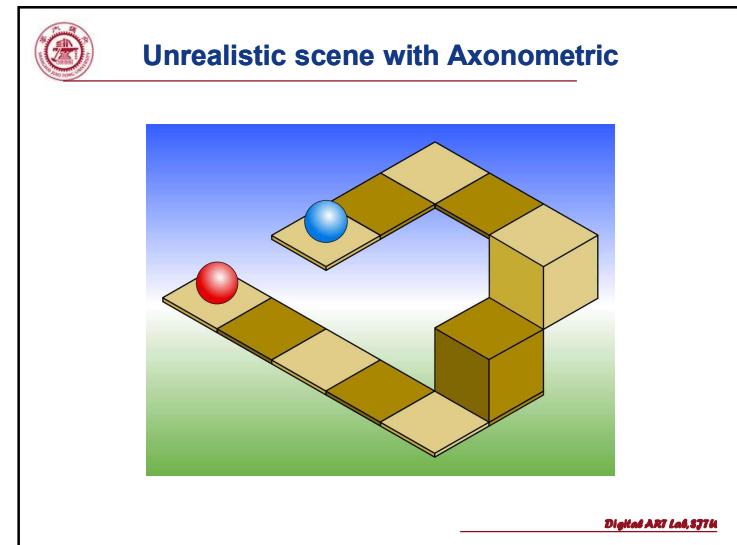
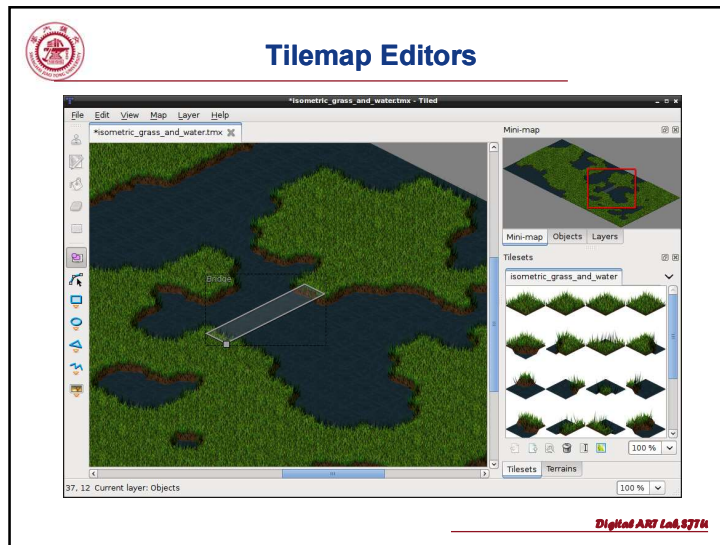
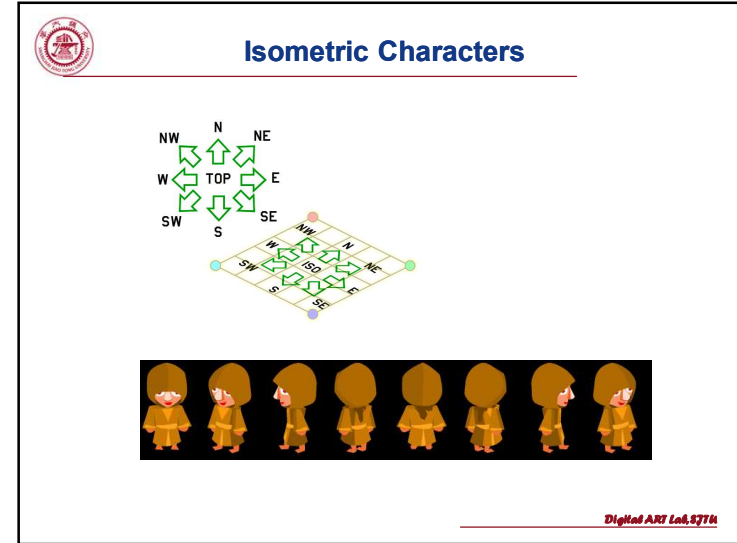
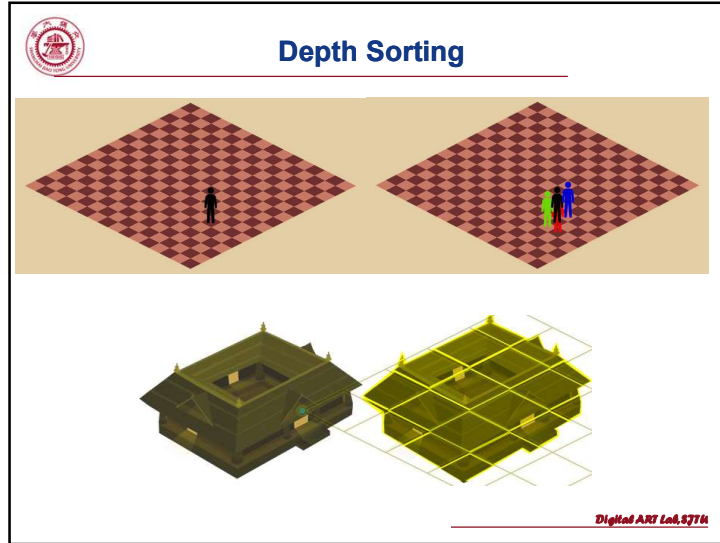
1 //Isometric to Cartesian:
2
3 cartX = (2 * isoY + isoX) / 2;
4 cartY = (2 * isoY - isoX) / 2;

```

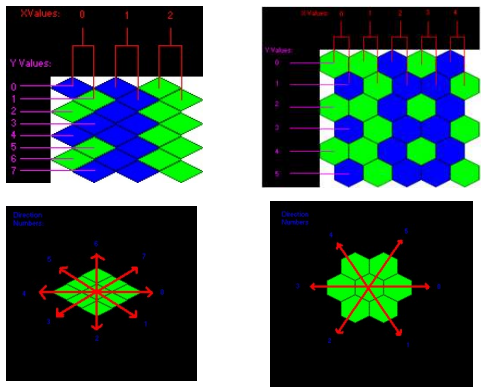
DEFAULT LEVEL WITH TRANSPARENT ISO GRID

GRASS ISO TILE WALL ISO TILE

Digital ART Lab, SZTH



Iso Maps vs. Hex Maps



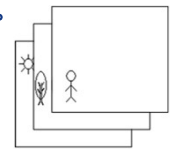
Iso Maps: The left map shows a diamond grid with values 0, 1, 2. The distance map shows 8 directions (N, NE, E, SE, S, SW, W, NW) with distances 1 through 8.

Hex Maps: The right map shows a hexagonal grid with values 0, 1, 2, 3, 4, 5, 6, 7. The distance map shows 6 directions (N, NE, E, SE, S, SW) with distances 1 through 6.

Digital ART Lab, SZTH

多层次地图

- 以下情况，可以考虑使用多层次地图。
 - 需要小地图能重叠或者有层次关系
 - 在背景上有多个物体运动
 - 需要模拟物体远近不同的透视关系
- 多层次地图的实现思想：
 - 在滚屏地图的基础上设置多个层次的地图
 - 设从底往上分别为0层，1层，...n层，把地图数据数组改为三维数组

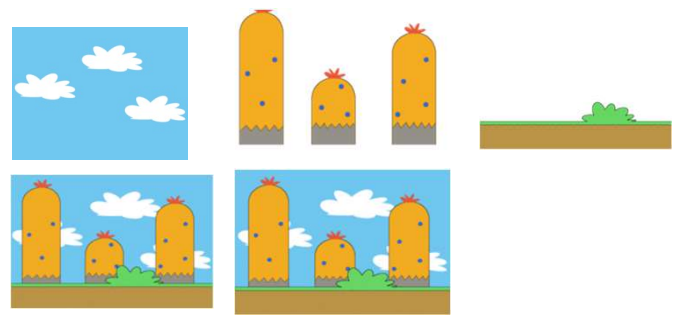


Digital ART Lab, SZTH



Digital ART Lab, SZTH

视差卷轴(Parallax Scrolling)



视差卷轴技术使每个图层以不同的速度运动，模拟景物远近不同的层次感。

Digital ART Lab, SZTH



地图的过程化生成 (Procedural Generation)

- With procedurally created maps, you can make sure that no two plays of your game are the same
- You can use various inputs, such as time or the current level of the player to ensure that the content changes dynamically even after the game has been built



Digital ART Lab, SZTU

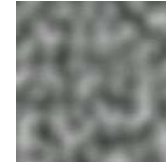


Procedural Generation

Perlin Noise

- gradient noise
- pseudo-random

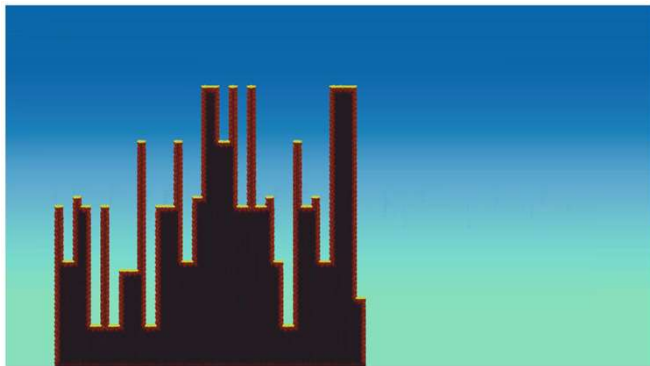
`Mathf.PerlinNoise(x, y)`



Digital ART Lab, SZTU



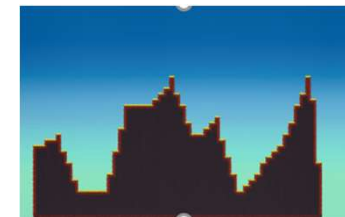
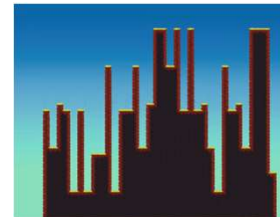
Example Result: Perlin Noise




Digital ART Lab, SZTU



Perlin Noise + Smooth

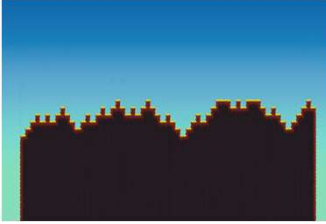
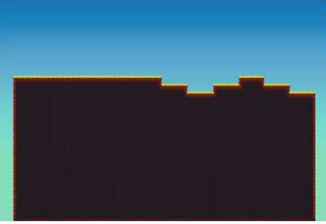


Digital ART Lab, SZTU



Random Walk


- ⊕ Flipping a coin (+1 or -1)

Random Walk

Random Walk Smoothed

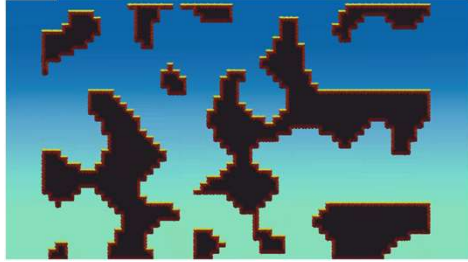
Digital ART Lab, SZTH




Procedural Generation

- ⊕ Perlin Noise to create a cave
 - Perlin noise value, which takes in the parameters of position(x,y) multiplied by a modifier value between [0,1] (larger means messier)
 - Round the value to either 0 or 1

```
newPoint = Mathf.RoundToInt(Mathf.PerlinNoise(x * modifier, y * modifier));
```



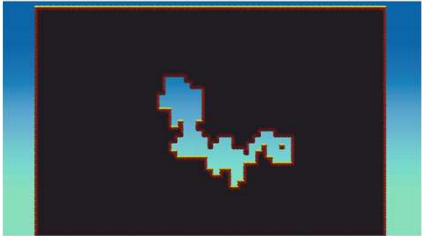
Digital ART Lab, SZTH




Random Walk

Random number: left, right, up, down (or +diagonal)

- ⊕ getting a random direction
- ⊕ move position and remove the tile
- ⊕ continue until reach the required amount to remove

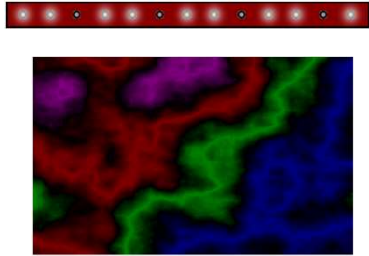


Digital ART Lab, SZTH

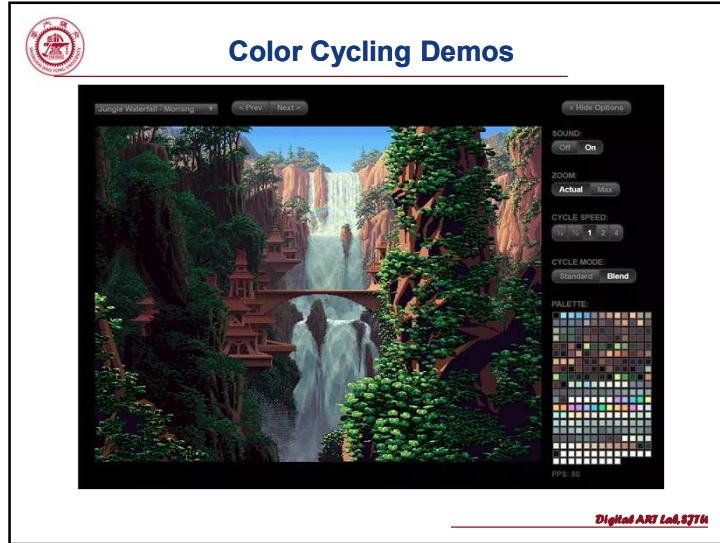


颜色轮换 (Color Cycling)

又称 palette shifting, 一种2D动态特效技术



Digital ART Lab, SZTH



精灵动画 (Sprite)

- ④ 精灵(sprite)技术: 前景是图像, 背景是透明的
- ④ 精灵动画(sprite animation): 将上一帧中精灵出现的地方用背景填充, 并在新的指定地点绘制精灵





透明区域

Digital ART Lab, SZTH

精灵动画 (续): 图像镂空

- ④ 将掩码图和背景图案进行按位AND, 使得原始图像的对应位置变空。
- ④ 将原始图像和上一步处理结果按位OR。
- ④ 结果: 原始图像贴到背景上并遮盖背景, 其余部分 (掩码图中白色部分) 没有贴到背景上。

$$(I_{mask} \text{ AND } I_{bg}) \text{ OR } I_{fg}$$

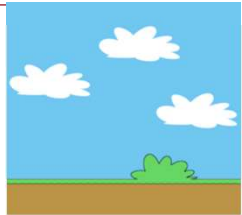
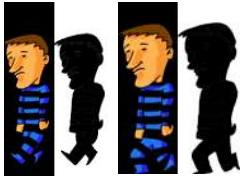



原始图像 掩码图

Digital ART Lab, SZTH

精灵动画 (续)

- ④ 实现
 - 一幅背景图
 - 一组模板图 (mask)
 - 人物的连续显示方式
 - 双缓冲机制
 - 不要在窗口中直接贴图, 避免闪烁
 - 建立一个内存缓存区
 - 所有的贴图动作都在这个缓存区上进行
 - 结果显示到操作窗口中

Digital ART Lab, SZTH



精灵动画（续）

- 对动画序列中的每一帧
 - Load 背景图
 - 确定sprite绘画的位置
 - 将某一掩码图与背景图作AND运算
 - 将对应人物图与前面结果作OR运算
 - 更新sprite绘画的位置



Digital ART Lab, SZTH



碰撞检测

- 对运动物体的碰撞判断是许多游戏程序中不可或缺的要素
- 常见的碰撞检测方法
 - 区域检测
 - 碰撞点检测
 - 颜色检测：较为精确，相对耗时



区域检测



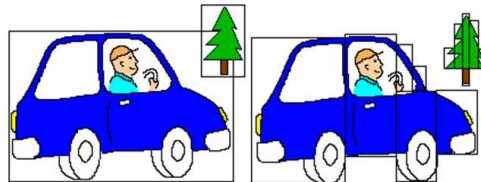
碰撞点检测

Digital ART Lab, SZTH



区域检测

- 采用某种规则形状逼近物体
- 物体之间的碰撞检测转化为规则形状之间的检测



Digital ART Lab, SZTH



碰撞点检测

- 本质是区域检测的一种
- 一般在两个运动物体中的一个物体上设置碰撞点，在另一个物体上设置检测区域，运行时逐个判断碰撞点是否在检测区域中。



Digital ART Lab, SZTH



颜色检测

- ④ 为树林做一张掩码图，将树林用黑色填充。
- ④ 为生成汽车驶入树林后面的效果：在背景上贴上汽车的图像，后在上面用镂空图技术画上树林。
- ④ 判断汽车图像在树林图像上的相对位置：将汽车图像上的点和掩码图上相应位置的点做按位AND操作，检查结果中是否有黑色点（RGB值为0）存在。任何颜色的RGB值与黑色图形进行按位AND运算，将得到黑色。如果存在黑色点，表明有碰撞。



Digital ART Lab, SZTH



2D Lighting



知乎 @Yumir

Digital ART Lab, SZTH



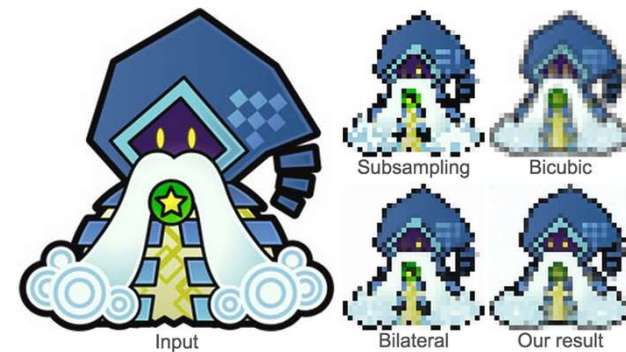
Depixelization



Digital ART Lab, SZTH



Adaptive Downsampling



Digital ART Lab, SZTH