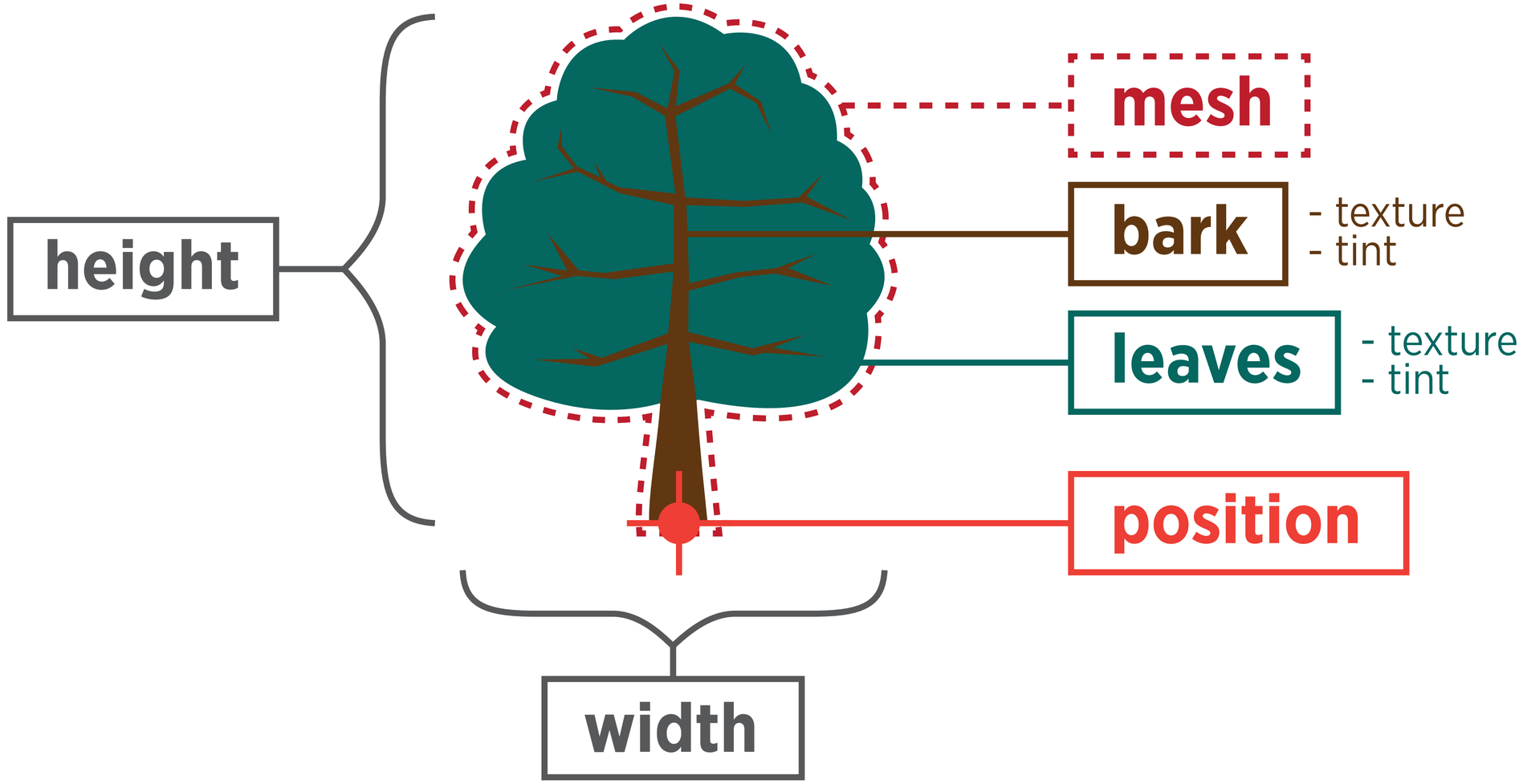


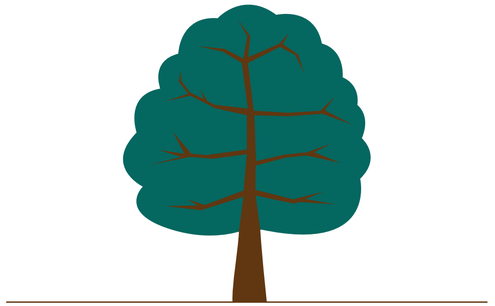
FlyWeight
Design Pattern

BCO 653 • Başar Emre ANIL




height
mesh
bark
- texture
- tint
leaves
- texture
- tint
position
width

```
class Tree
{
  private:
    Mesh mesh_;
    Texture bark_;
    Texture leaves_;
    Vector position_;
    double width_;
    double height_;
    Color barkTint_;
    Color leafTint_;
};
```



mesh

bark

leaves

position

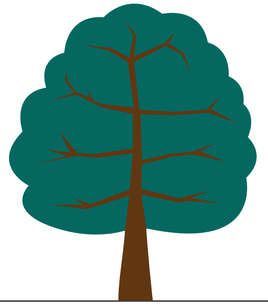
width

height

barkTint

leafTint

```
class Tree
{
  private:
    Mesh mesh_;
    Texture bark_;
    Texture leaves_;
    Vector position_;
    double width_;
    double height_;
    Color barkTint_;
    Color leafTint_;
};
```



mesh

bark

leaves

position

width

height

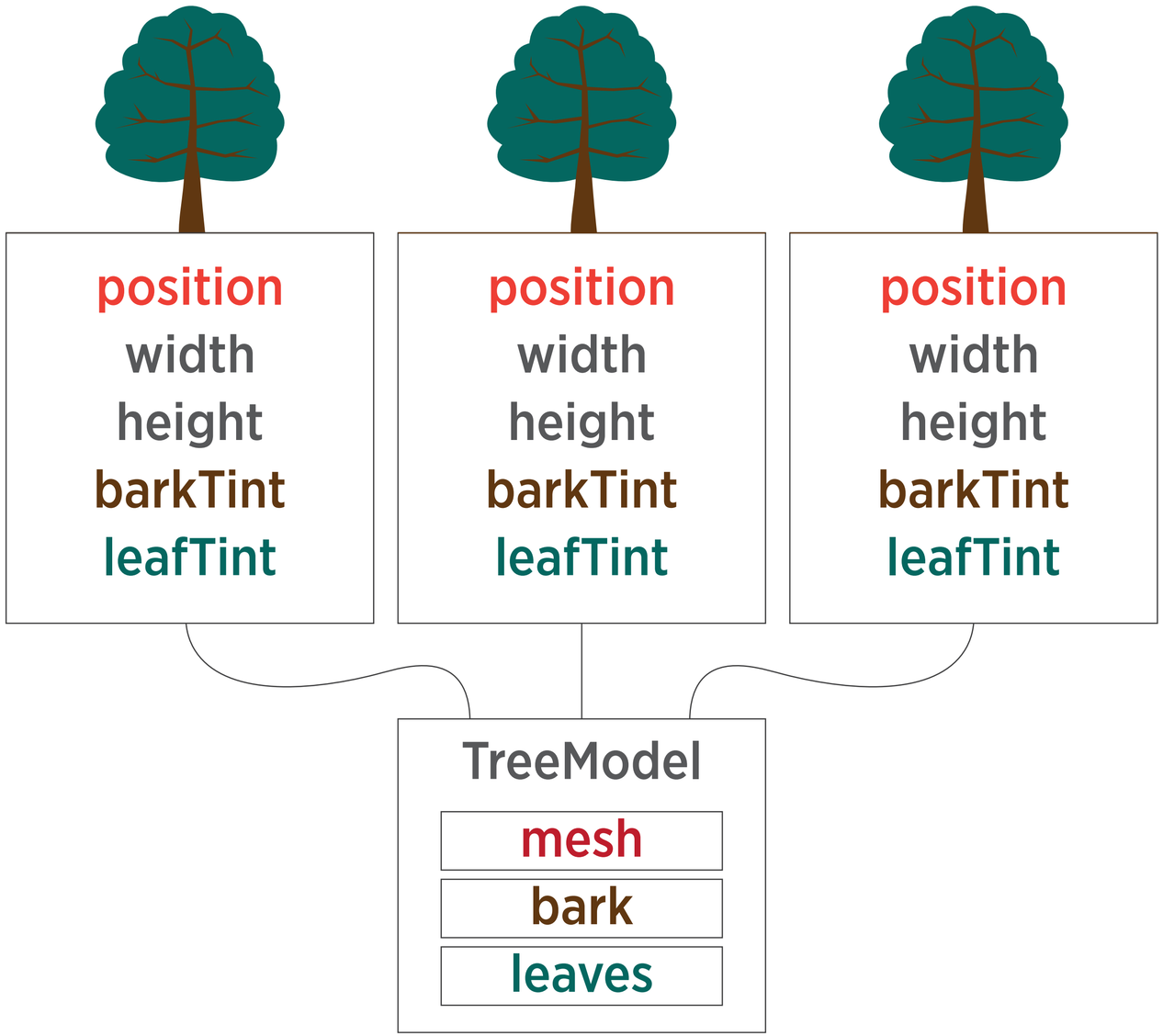
barkTint

leafTint

```
class TreeModel
{
  private:
    Mesh mesh_;
    Texture bark_;
    Texture leaves_;
    Texture leaves_;
};

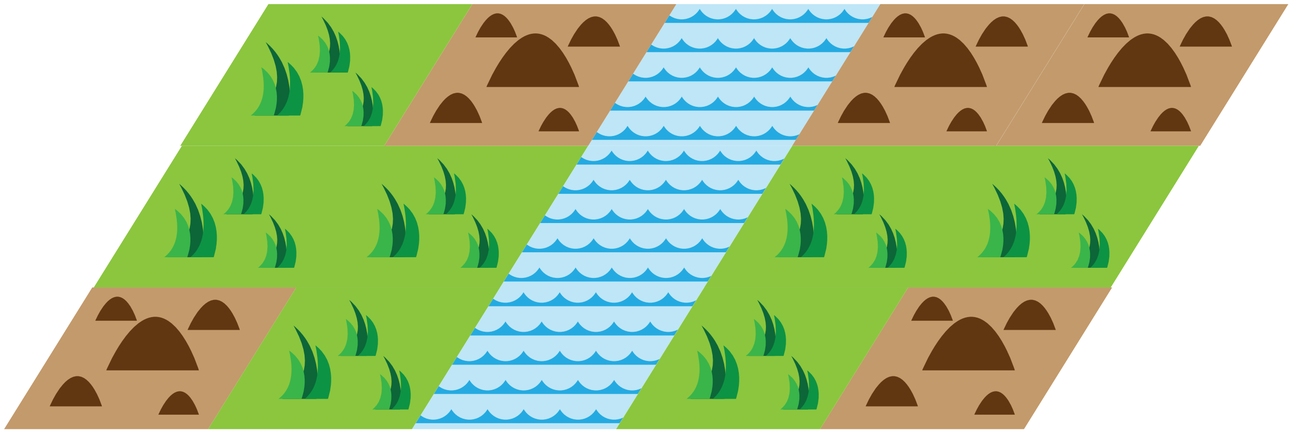
class Tree
{
  private:
    TreeModel* model_;

    Vector position_;
    double height_;
    double thickness_;
    Color barkTint_;
    Color leafTint_;
};
```



```
enum Terrain
{
  TERRAIN_GRASS,
  TERRAIN_HILL,
  TERRAIN_RIVER
  // Other terrains...
};

class World
{
  private:
    Terrain tiles_[WIDTH][HEIGHT];  // tiles_[5][3]
};
```



```
int World::getMovementCost(int x, int y)
{
  switch (tiles_[x][y])
  {
    case TERRAIN_GRASS: return 1;
    case TERRAIN_HILL: return 3;
    case TERRAIN_RIVER: return 2;
    // Other terrains...
  }
}

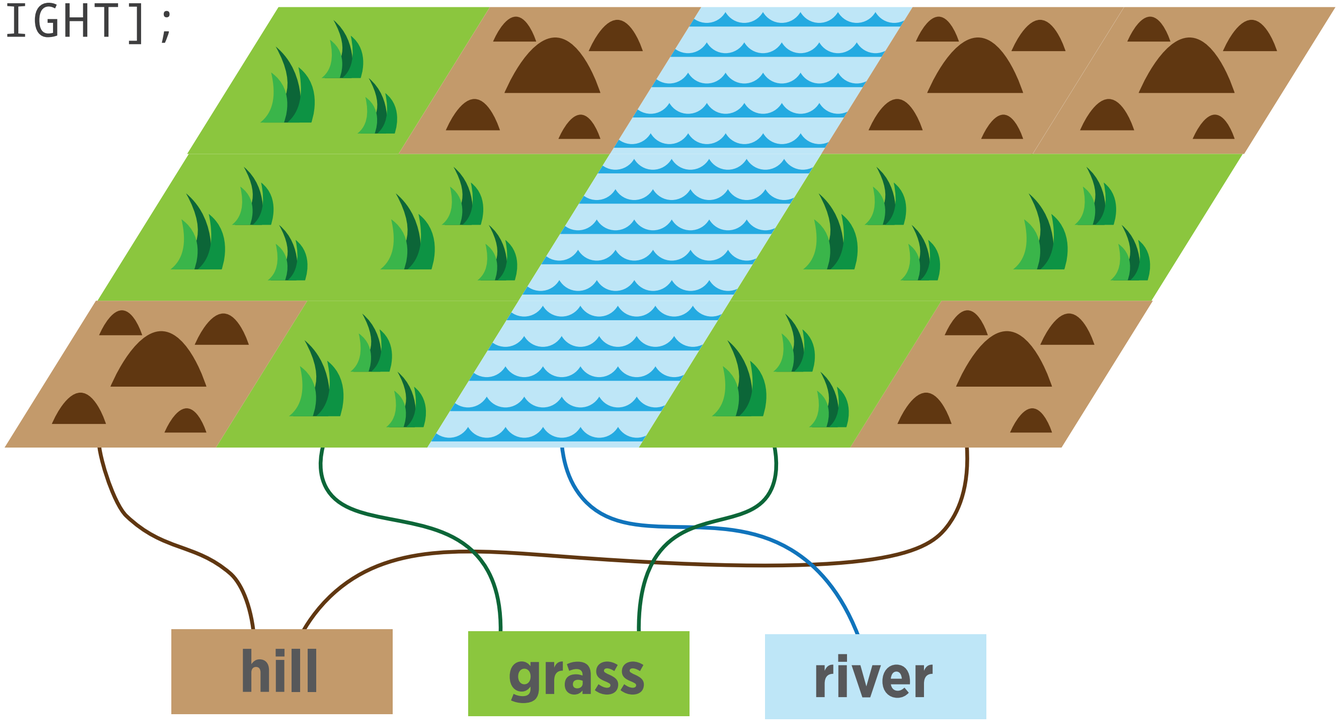
bool World::isWater(int x, int y)
{
  switch (tiles_[x][y])
  {
    case TERRAIN_GRASS: return false;
    case TERRAIN_HILL: return false;
    case TERRAIN_RIVER: return true;
    // Other terrains...
  }
}
```

```
class Terrain
{
  public:
    Terrain(int movementCost,
            bool isWater,
            Texture texture)
      : movementCost_(movementCost),
        isWater_(isWater),
        texture_(texture)
      {}

  int getMovementCost() const { return movementCost_; }
  bool isWater() const { return isWater_; }
  const Texture& getTexture() const { return texture_; }

  private:
    int movementCost_;
    bool isWater_;
    Texture texture_;
};
```

```
class World
{
  private:
    Terrain* tiles_[WIDTH][HEIGHT];
    // Other stuff...
};
```



```
class World
{
  public:
    World()
    : grassTerrain_(1, false, GRASS_TEXTURE),
      hillTerrain_(3, false, HILL_TEXTURE),
      riverTerrain_(2, true, RIVER_TEXTURE)
      riverTerrain_(2, true, RIVER_TEXTURE)
    {}

private:
    Terrain grassTerrain_;
    Terrain hillTerrain_;
    Terrain riverTerrain_;
    // Other stuff...
};
```

```
void World::generateTerrain()
{
  // Fill the ground with grass.
  for (int x = 0; x < WIDTH; x++)
  {
    for (int y = 0; y < HEIGHT; y++)
    {
      // Sprinkle some hills.
      if (random(10) == 0){
        tiles_[x][y] = &hillTerrain_;
      }
      else{
        tiles_[x][y] = &grassTerrain_;
      }
    }
  }
  // Lay a river.
  int x = random(WIDTH);
  for (int y = 0; y < HEIGHT; y++) {
    tiles_[x][y] = &riverTerrain_;
  }
}
```

```
const Terrain& World::getTile(int x, int y) const
{
  return *tiles_[x][y];
}

int MovCost = world.getTile(2, 3).getMovementCost();
```

```
for (int x=0; x<countPeople); x++)
{
  cout << "ThAnK YoU!!." << endl;
}
```