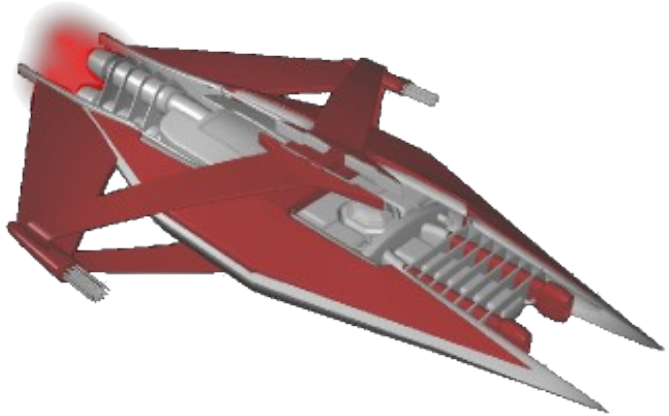
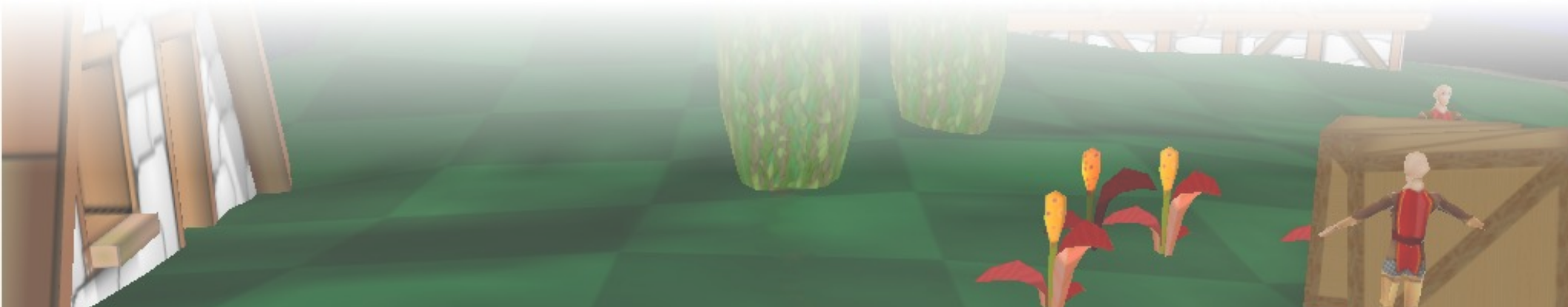


Multimediaentwicklung unter Linux

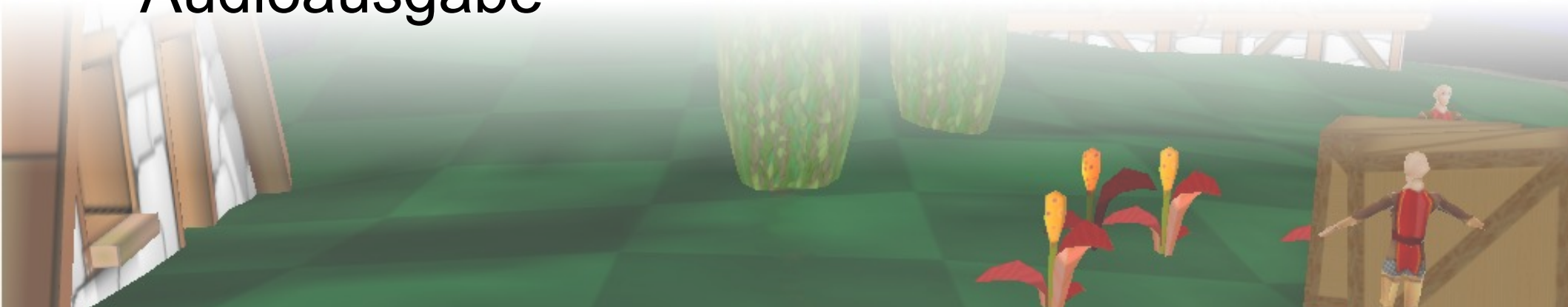
Inhalt

- Aller Anfang ist einfach
- Auf in die dritte Dimension
- Vernetzung
- Obey gravity, It's the law!
- Scripting
- Alles zusammen



Erste Schritte

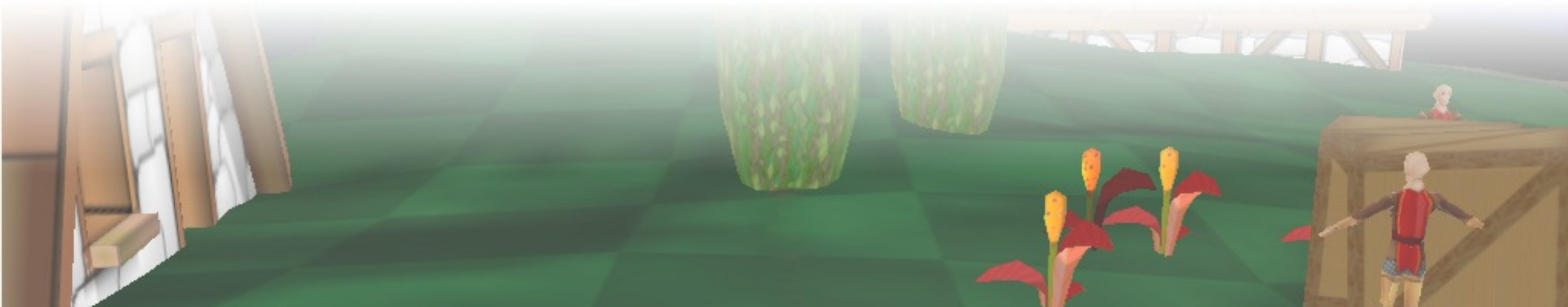
- Vorausetzung (gcc oder bindings)
- Ein Fenster öffnen
- 2D Grafik
  - Einfach Operationen [Linien/Punkte]
  - Bilder laden und darstellen
- Eingaben verarbeiten [Maus/Tastatur/...]
- Audioausgabe



Ein Fenster öffnen

aller Anfang ist einfach

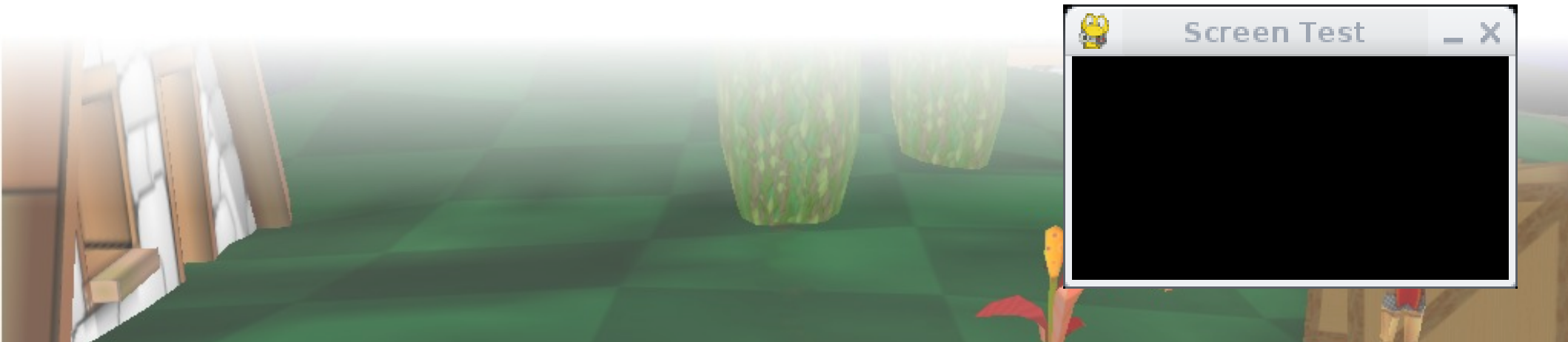
- Initialisieren des Fensters
  - Setzen von Parametern
    - DoubleBuffering
    - Hardware Acceleration
    - Fullscreen
  - Öffnen und erhalten eines Buffers zum darstellen



Ein Fenster öffnen

aller Anfang ist einfach

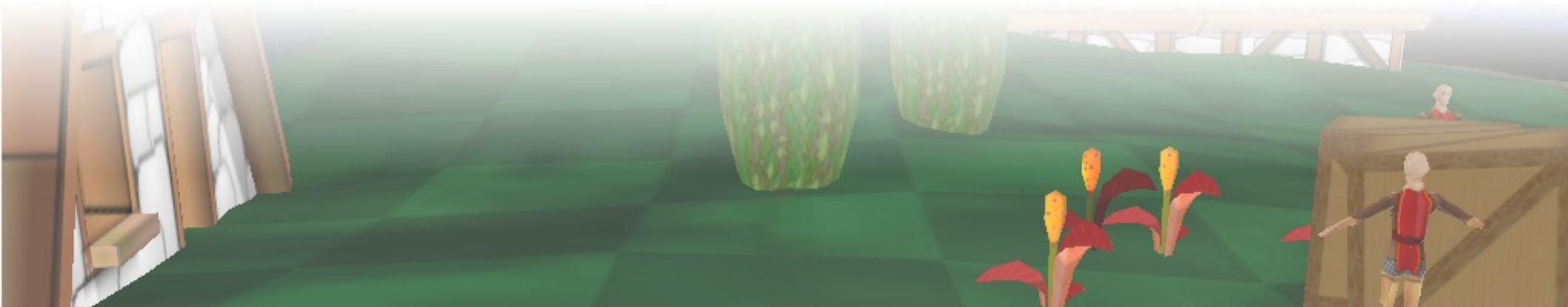
```
# init the PyGame Modules (similar to SDL but there
you
# have to do it for each module)
pygame.init()
# create the window with the size 640x480, this will
# create a window and not a fullscreen image
window = pygame.display.set_mode((640, 480))
# set Caption of window
pygame.display.set_caption('Screen Test')
```



2D Grafik

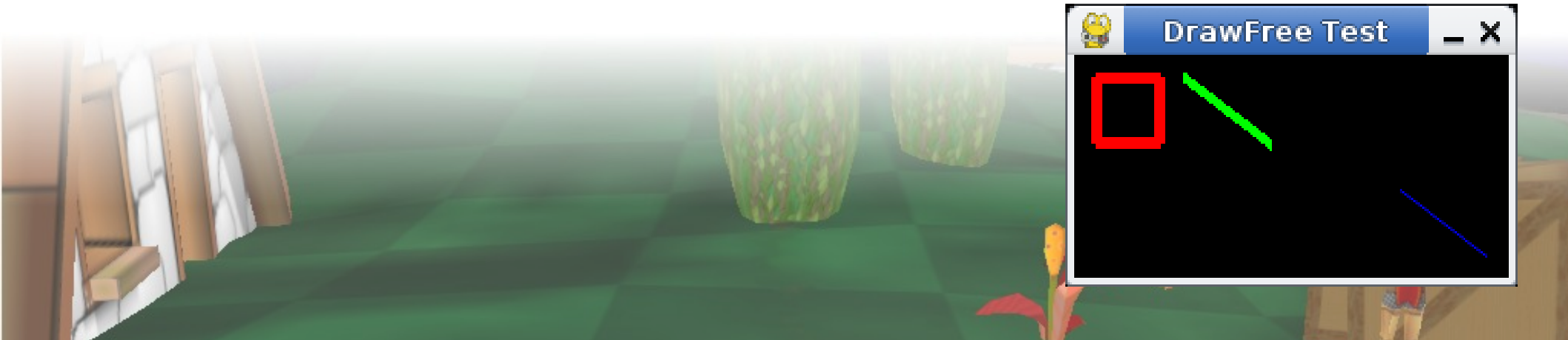
aller Anfang ist einfach

- Linien / Rechteck / Punkt
  - Zeichnen in einen Buffer
- Bilder
  - Laden,
  - Zeichnen und
  - Bewegen



2D Grafik


```
# draw a red rect on the screen position: 10, 10
# size: 30, 30 width: 5
pygame.draw.rect(screen, (255,0,0), (10,10, 30, 30), 5)
# draw a green line on the screen start position: 50, 10
# end postion: 90, 40 width: 5
pygame.draw.line(screen, (0,255,0), (50,10) , (90, 40), 5)
# draw a blue antialiazed line on the screen
# start position: 150, 110 end postion: 190, 140
pygame.draw.aaline(screen, (0,0,255), (150,60) , (190,
  90))
# flip the screen that the new surface get shown
pygame.display.flip()
```



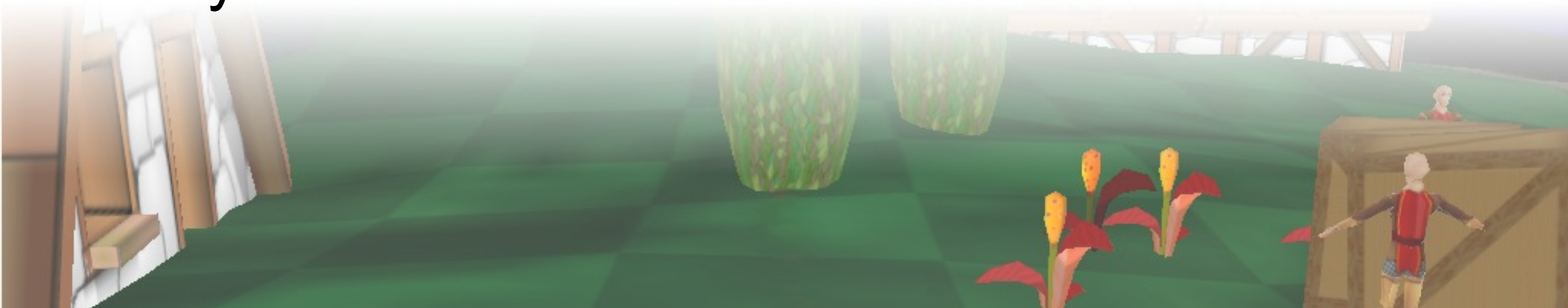
2D Grafik


```
# get correct path to the image file
monster_filename = os.path.join("data",
   "SpaceInvader.png")
# load monster image
monster_surface = pygame.image.load(monster_filename)

# blit the image to the surface on the position 10, 10
screen.blit(monster_surface, (10, 10))
# flip the screen that the new surface get shown
pygame.display.flip()
```

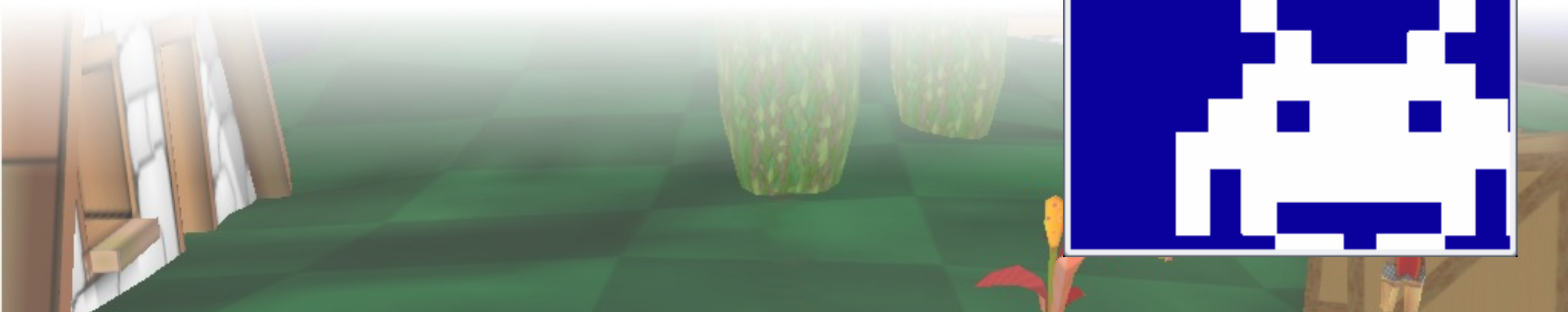
Input

- Libraries
  - SDL
  - PyGame
  - OIS
- Keyboard
- Mouse
- Joystick



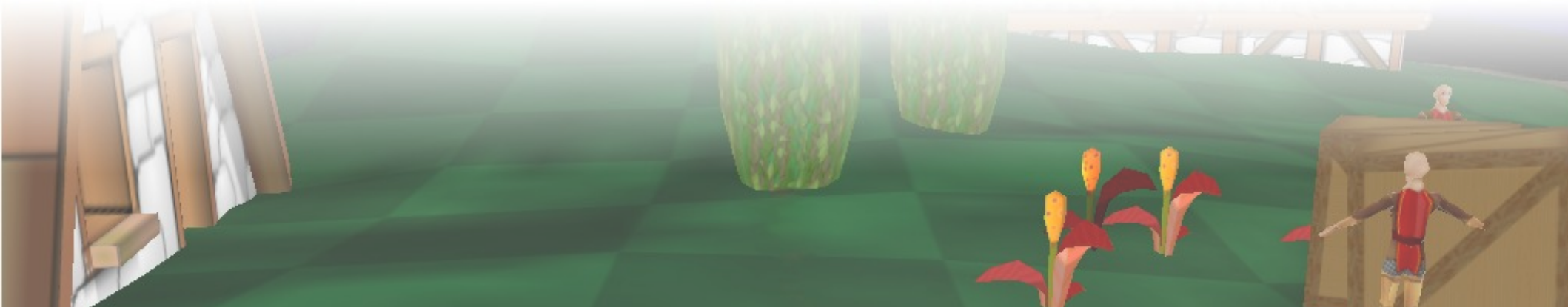
Input


```
event = pygame.event.get()
 for event in events:
     if event.type == QUIT:
         return False
     # handle Keydown events
     elif event.type == KEYDOWN:
         if event.key == K_LEFT:
             # Move left
             monster_move[0] -= 1
```



Audio Ausgabe

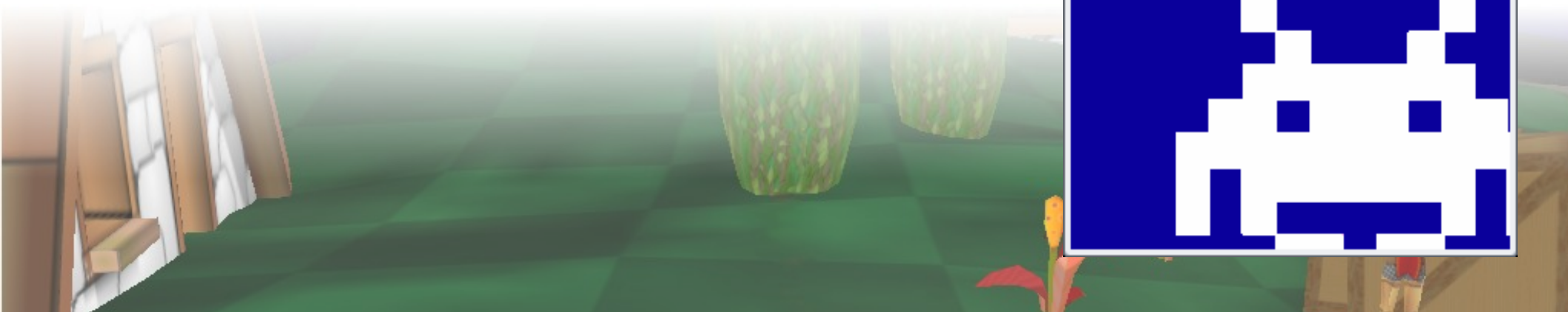
- Effekte
  - Wave
- Musik
  - Mp3
  - Ogg



Audio Ausgabe

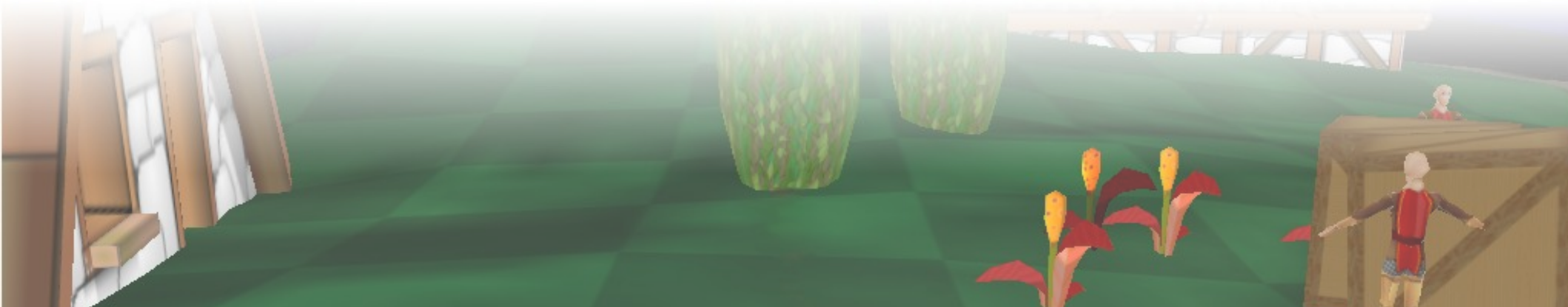
aller Anfang ist einfach

```
file_sound = os.path.join("data", "pling.wav")
pling_sound = pygame.mixer.Sound(file_sound)
pling_sound.play()
```



Auf in die dritte Dimension

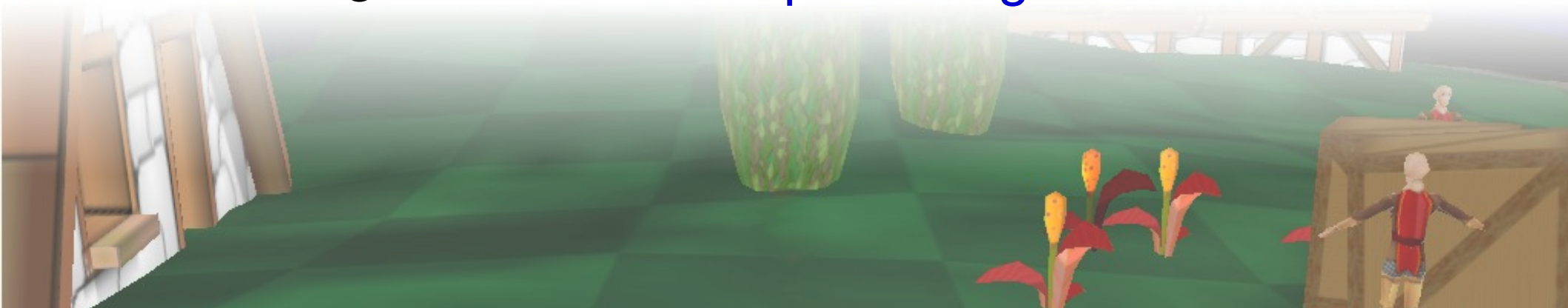
- OpenGL
  - Einzelne Polygone
  - Komplexe Objekte
- Modelle
- Engines
- Soundeffekte in 3D



OpenGL

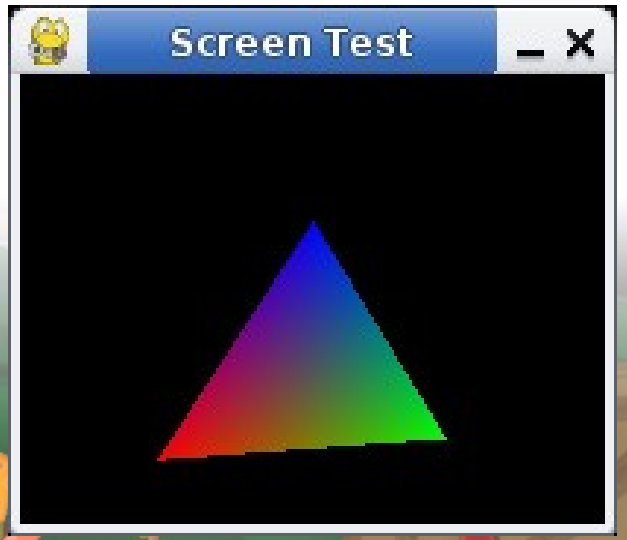
auf in die dritte Dimension

- Erstellen eines OpenGL-Fensters
- Einfache Polygone
  - Zeichnen
  - Bewegen im Raum
- Komplexe Objekte
  - Zeichnen
  - Bewegen im Raum http://nehe.gamedev.net/



OpenGL


```
# start with drawing triangles
glBegin(GL_TRIANGLES)
# set color of vertex
glColor3fv((0.0, 0.0, 1.0))
# draw vertex with the set color
glVertex3fv((0.0, 0.5, 0.0))
glColor3fv((0.0, 1.0, 0.0))
glVertex3fv((1.0, -1.0, 0.0))
glColor3fv((1.0, 0.0, 0.0))
glVertex3fv((-1.0, -1.0, 0.0))
# end drawing triangles
glEnd()
```



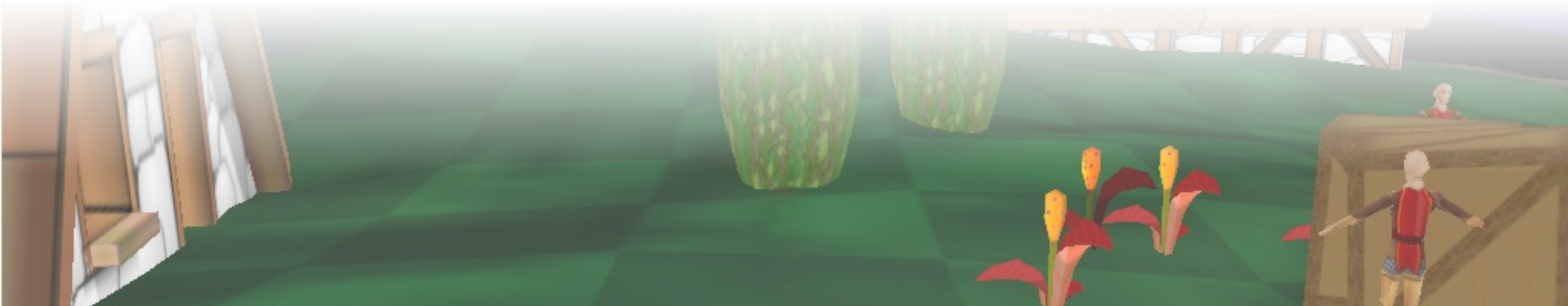
OpenGL


```
CUBE_POINTS = (
    (0.5, -0.5, -0.5), (0.5, 0.5, -0.5),
    (-0.5, 0.5, -0.5), (-0.5, -0.5, -0.5),
    (0.5, -0.5, 0.5),  (0.5, 0.5, 0.5),
    (-0.5, -0.5, 0.5), (-0.5, 0.5, 0.5)
)
...
glRotatef(1, 0, 1, 0)
...
```

3D Modelle

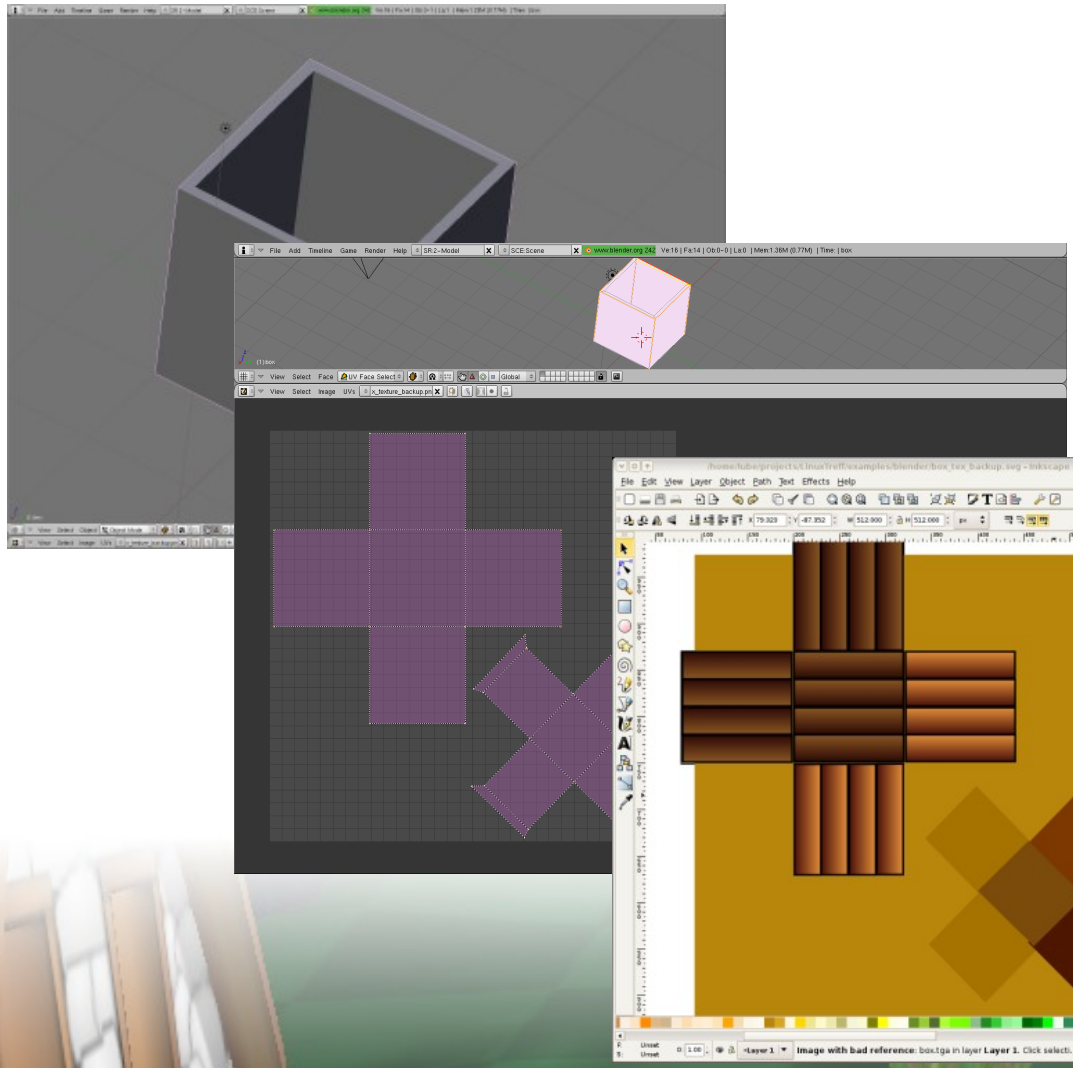
auf in die dritte Dimension

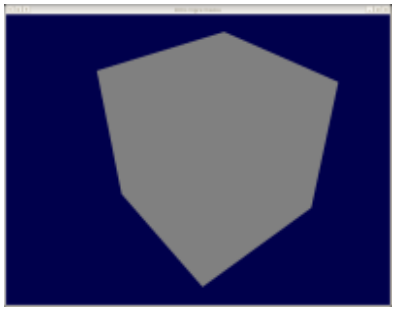
- Erstellen von 3D-Modellen
  - Blender
  - Max 3D Studio
  - Maya
- Erstellen von Texturen
  - Bemalen des 3D Models
  - Gimp / Inkscape

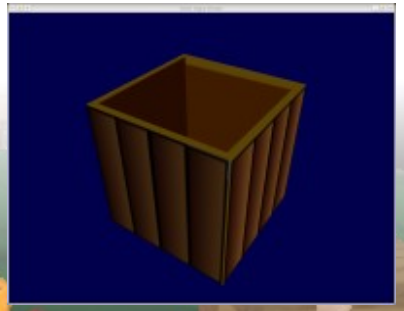


3D Modelle

auf in die dritte Dimension







3D Engines

auf in die dritte Dimension

(http://www.ogre3d.org)

(http://irrlicht.sourceforge.net/)

(http://www.openscenegraph.com)

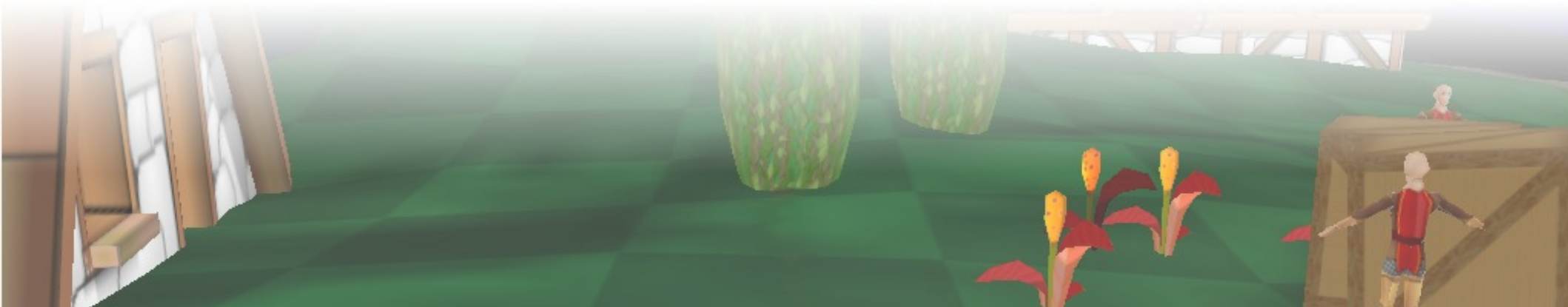
(http://www.panda3d.org)

(http://www.crystalspace3d.org)

Soundeffekte in 3D

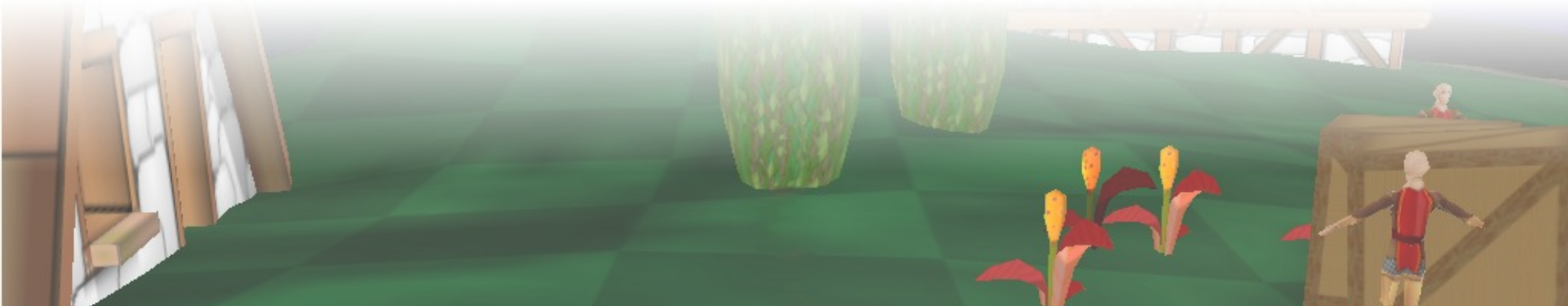
auf in die dritte Dimension

- Sound von einzelnen Objekten
- Bibliotheken
  - OpenAL
  - FMOD



Vernetzung

- Interaktivität
  - MMORPGs (WoW, GW)
- Networking (Low Latency)
  - UDP vs. TCP
- Synchronisierung von Clients und Server
  - Sicherheit
  - Flüssiges Verhalten



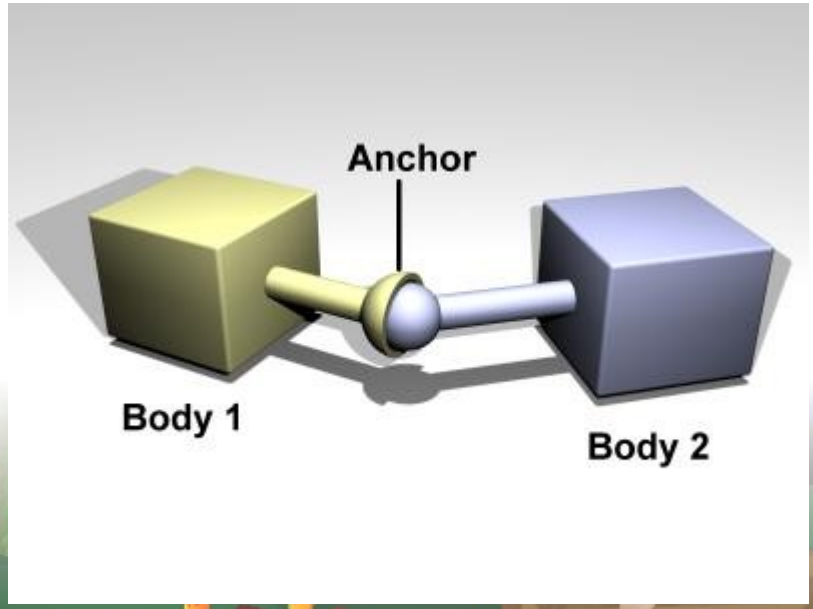
Vernetzung - Gut zu wissen

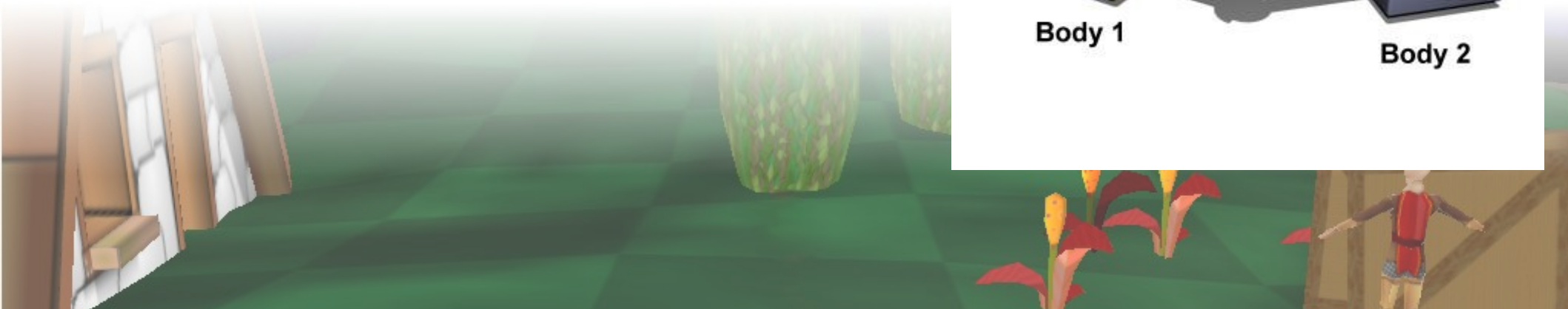
- Datenraten
  - 24 Frames/Sek, 20 bewegte Objekte, Position (3 Floats), Rotation (4 Floats) pro Objekt
  - = 105 kBit/Sek.

- Nagle Algo.
  - setsockopt(m_iSock, IPPROTO_TCP, **TCP_NODELAY**, (Char*)&iFlag, sizeof(Int));

Obey gravity, It's the law!

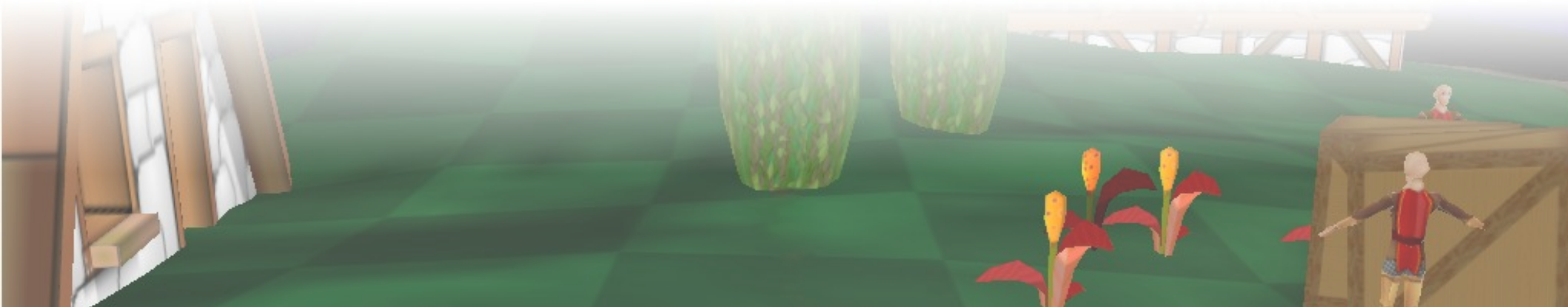
- Feedback - reelles verhalten in der virtuellen Welt
- Rechenaufwand vs. Genauigkeit
- Bibliotheken
  - ODE
  - Newton
  - PhysiX





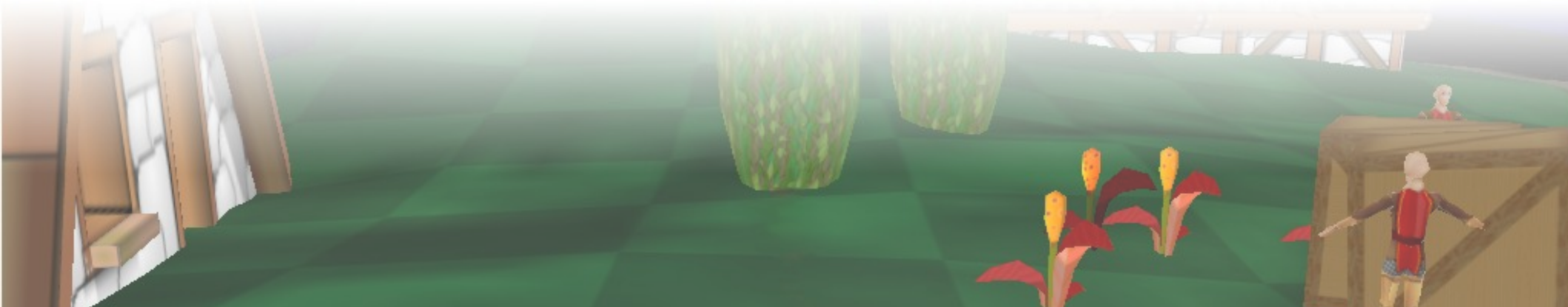
Scripting

- Warum?
  - Einfacher
  - Weniger Fehler
  - Weniger Code
  - Flexibler
  - Kein Compilieren



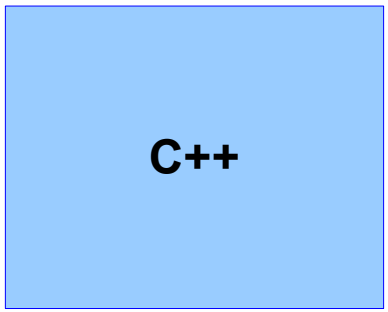
Scripting

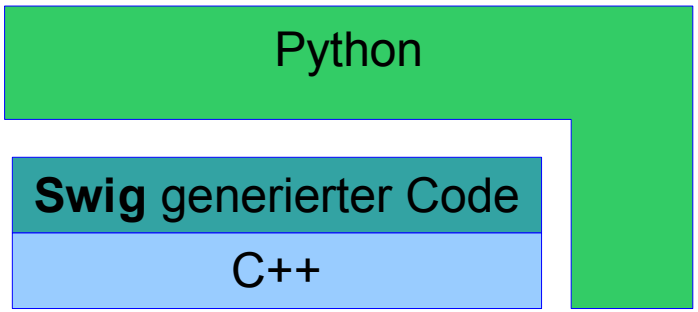
- Wieso nicht alles in der Scriptsprache?
  - Bottlenecks
- Wie verbinden von C++ and Scriptsprache
  - SWIG
  - Glue Code

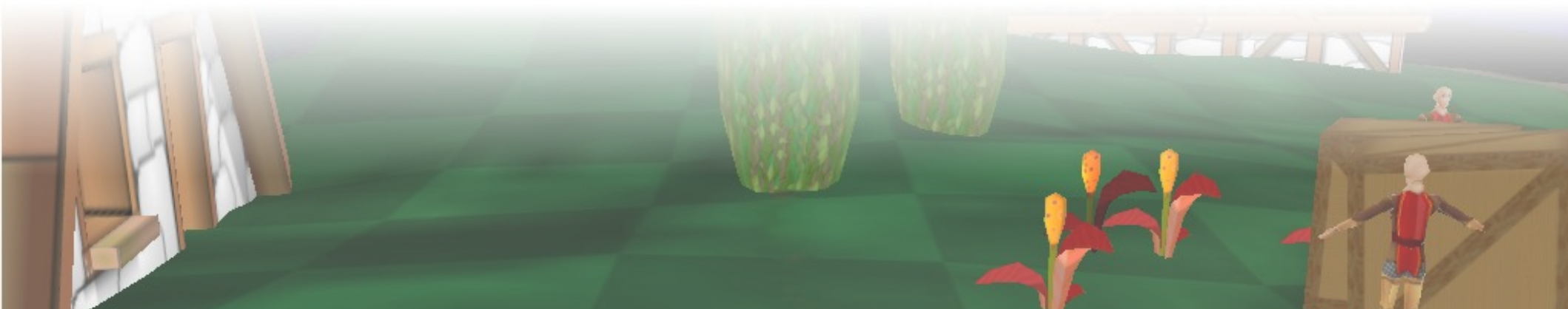


Scripting

- Beispiel: 30'000 Float Multiplikationen

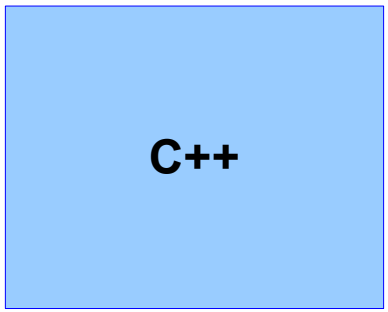






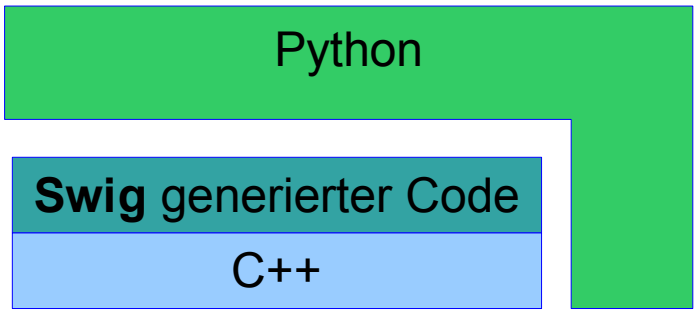
Scripting

- Beispiel: 30'000 Float Multiplikationen

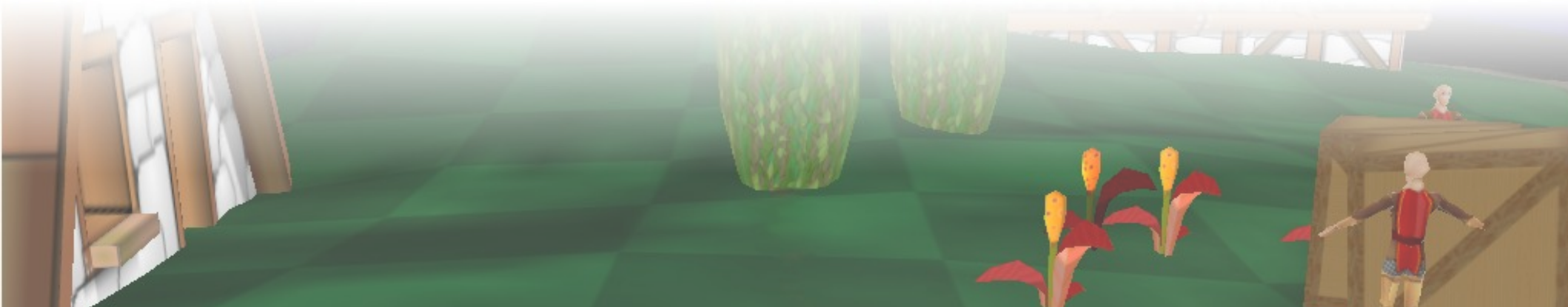


~0.1 Millisek.

~21.6 Millisek.

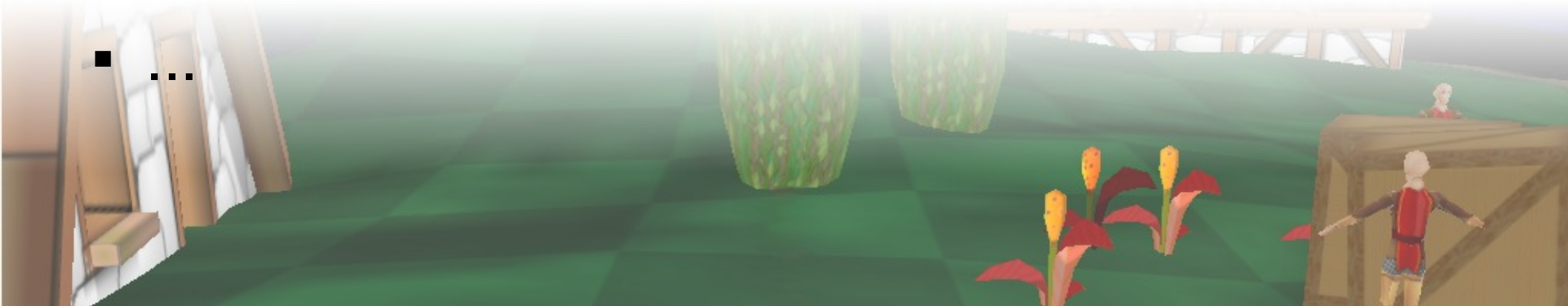


~0.12 Millisek.



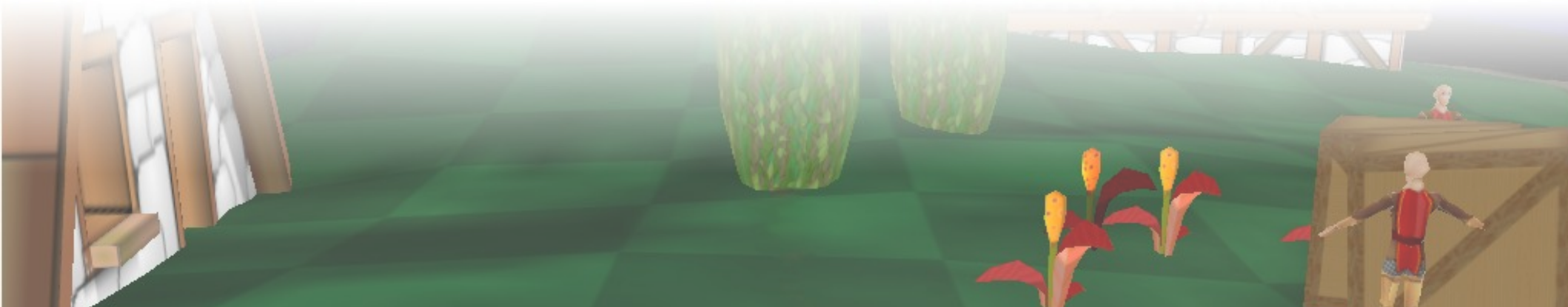
Scripting

- Python
  - Weit verbreitet
  - Objekt orientiert
- LUA
  - Nicht Objekt orientiert
  - Für Gamescripting entwickelt
  - Einfach Glue Code zu erstellen
- ...



Und jetzt alles zusammen

- Client/Server-Architektur
- ODE für physikalische Berechnungen
- Ogre zur Grafikausgabe
- OIS für die Verarbeitung von Eingaben
- Python als Scriptsprache



Links

- SDL (http://www.libsdl.org)
- PyGame (http://www.pygame.org)
- OIS (http://www.wreckedgames.com/wiki/index.php/WreckedLibs:OIS)
- Ogre (http://www.ogre3d.org)
- ODE (http://www.ode.org)
- Python (http://www.python.org)
- Lua (http://www.lua.org)
- OpenAL (http://www.openal.org)
- ClanLib (http://www.clanlib.org/)
- Allegro (http://alleg.sourceforge.net/index.de.html)
- Newton (http://www.newtondynamics.com/