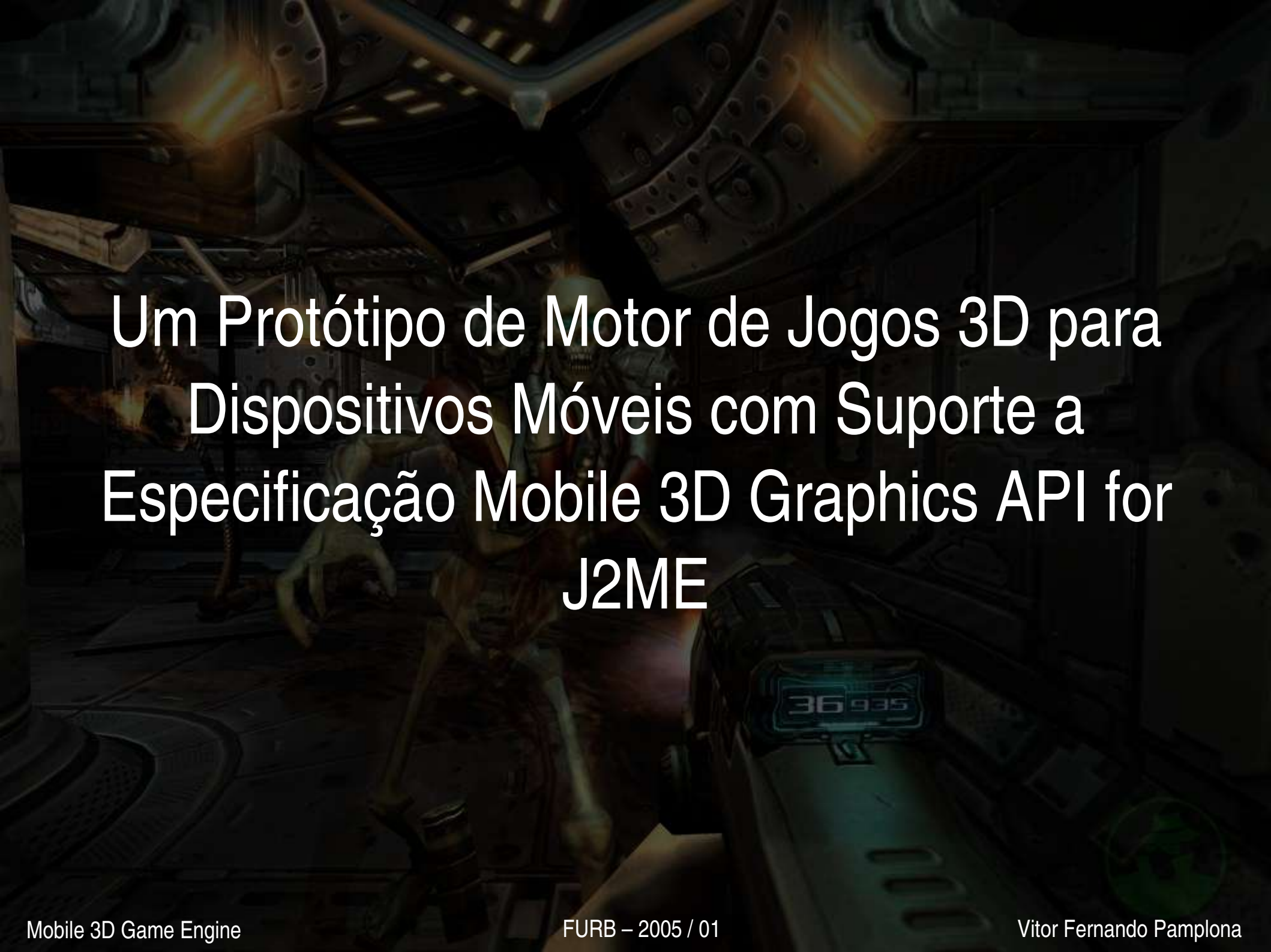




Comming soon...

36935





# Um Protótipo de Motor de Jogos 3D para Dispositivos Móveis com Suporte a Especificação Mobile 3D Graphics API for J2ME



# Roteiro

- Introdução
- Objetivos
- Fundamentação Teórica
- Desenvolvimento
- Conclusão

# Introdução

- Jogos, o entretenimento milenar
- Qualidade gráfica e complexidade computacional
- Celulares e poder computacional
- Jogos para celulares
- OpenGL ES e Mobile 3D Graphics API (M3G)

# Objetivos

- Adquirir experiência com desenvolvimento de jogos
- Analisar a tecnologia Java nos dispositivos móveis
- Verificar como os aparelhos se comportam com rotinas 3D

## Como alcançar

- Criar um motor de jogos 3D para celulares e uma aplicação exemplo, utilizando a M3G como base.



The background is a dark, industrial 3D game environment. In the center, a skeletal character with glowing blue eyes and a red, fleshy appendage is visible. To the right, a digital display shows the number '36935'. The scene is lit with warm, yellowish light from overhead fixtures.

# Fundamentação Teórica

# Jogos

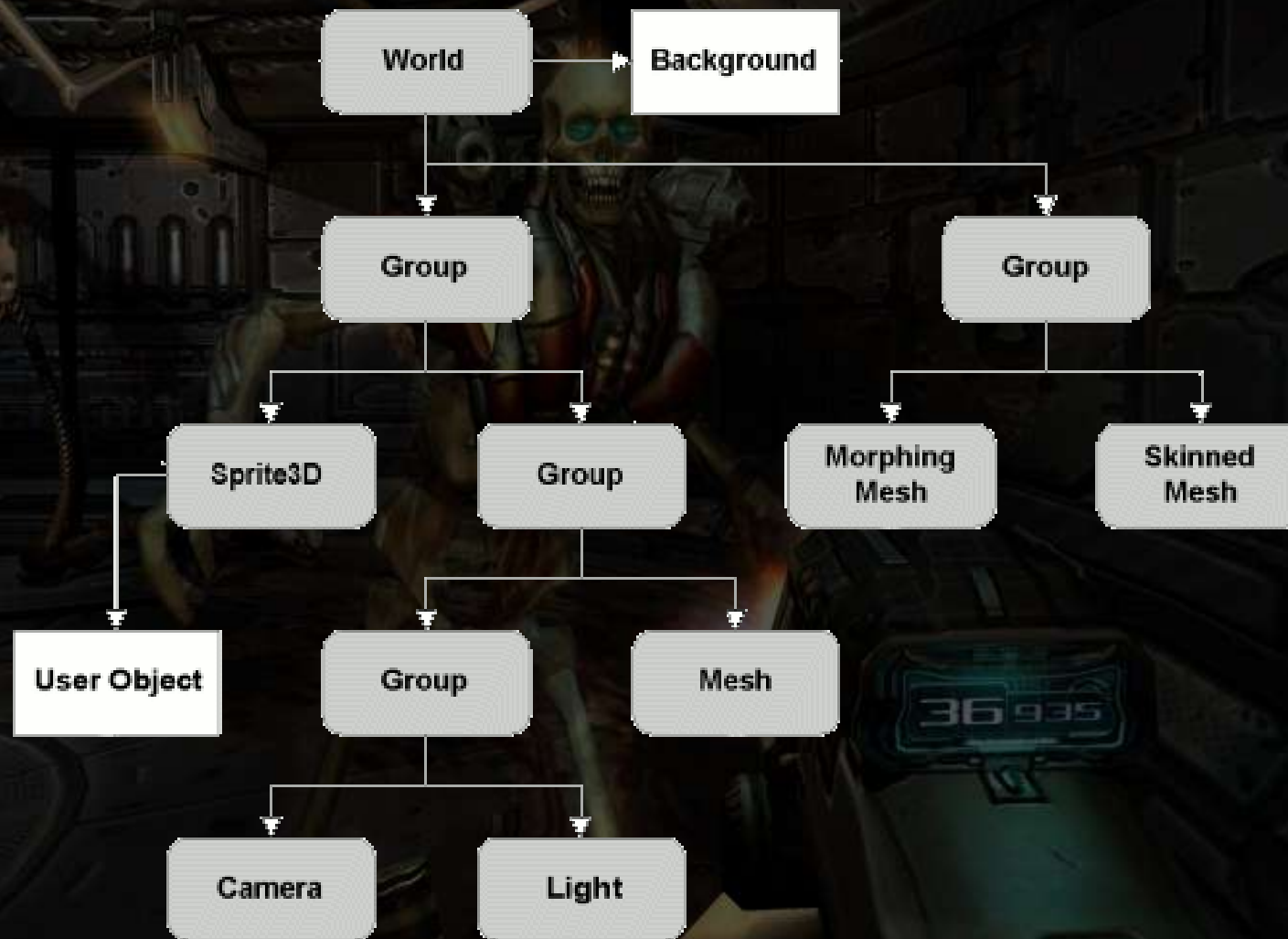
- Arquitetura de um Jogo:
  - Enredo, história, charadas e objetivos
  - Interface interativa, jogabilidade, gráficos e efeitos
  - Motor de jogos, o controle central do jogo

# Motor de jogos 3D





# Grafo de Cena



# Mobile 3D Graphics API for J2ME (M3G)

- Especificação de uma API 3D para celulares.
- JSR liderada pela Nokia e concluída em dez. de 2003
- Projetada para trabalhar sobre a OpenGL ES
- Define um arquivo de modelo 3D binário
- Possui rotinas otimizadas para leitura de imagens e modelos 3D
- Tamanho em ROM e RAM menor que 150KB



# O formato de arquivos Wavefront

- Formato padrão para armazenar modelos 3D
- Reconhecido mundialmente
- Todas as ferramentas de modelagem 3D possuem suporte ao formato
- Definido em um arquivo texto, não em binário
- Trabalha com objetos visíveis, ignorando os nós de Câmera, Luz, sub-grupo e com efeitos especiais
- Define arquivos OBJ para objetos e MTL para texturas

# Especificação de Arquivo Wavefront

Arquivo OBJ:

# <algum texto>

v <float> <float> <float>

vn <float> <float> <float>

vt <float> <float>

mtllib <arquivo>

g <nome do grupo>

usemtl < nome de um material>

f <int[/int[/int]] [int[/int[/int]] [int[/int[/int]]]>

Arquivo MTL:

newmtl <nome>

Ka <float> <float> <float>

Kd <float> <float> <float>

lIum [1,2]

Ks <float> <float> <float>

d <float>

Ns <float>

Tr <float>

map\_Kd <arquivo>



# Contexto Atual

- Existem jogos 3D profissionais
- O mundo só pensa em jogos quando o assunto é celular
- No Brasil, os jogos não são vistos com tanto entusiasmo
- Os celulares com suporte a M3G não são populares
- A documentação sobre a M3G API é escassa
- As operadoras ainda não permitem criar jogos on-line



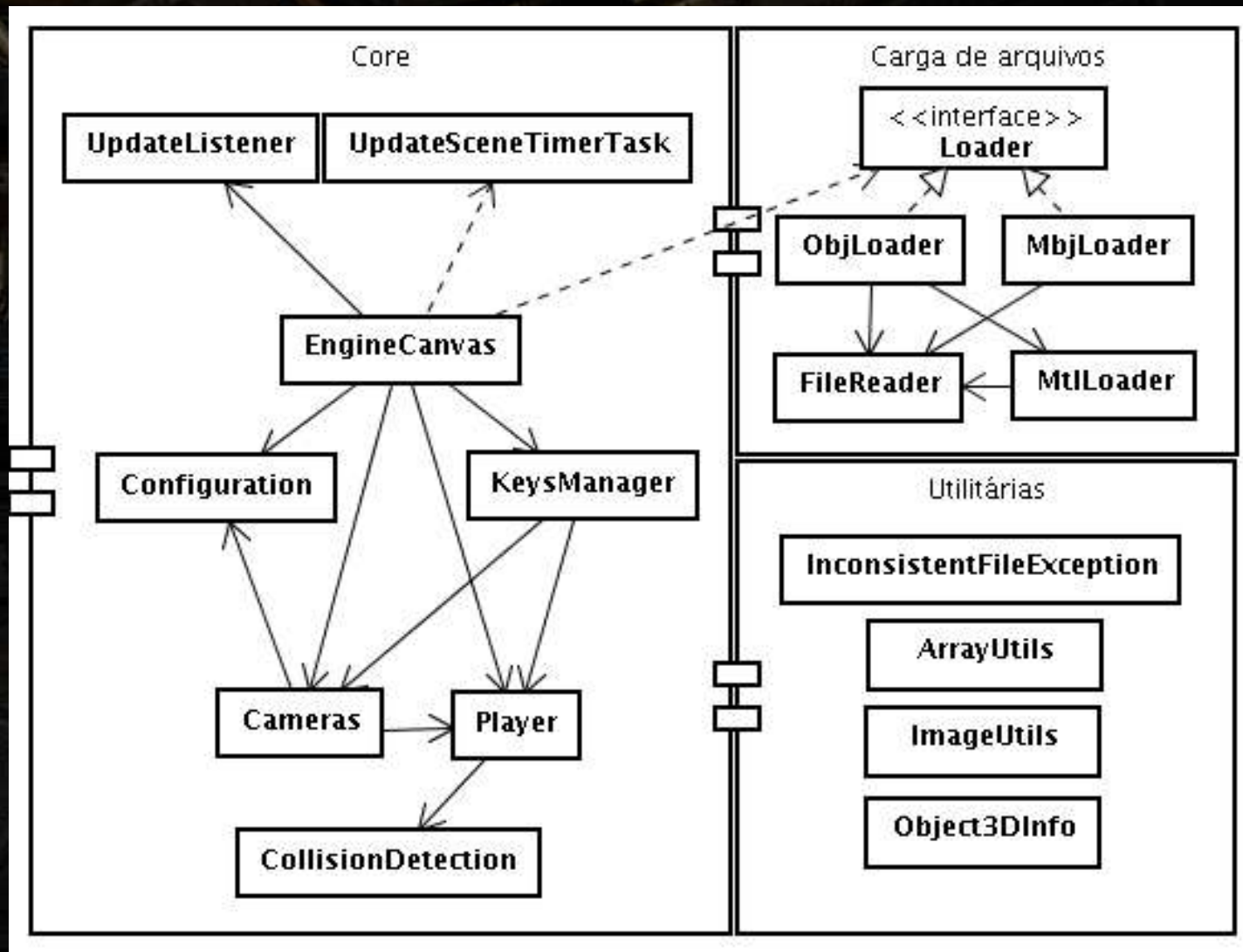
# Desenvolvimento



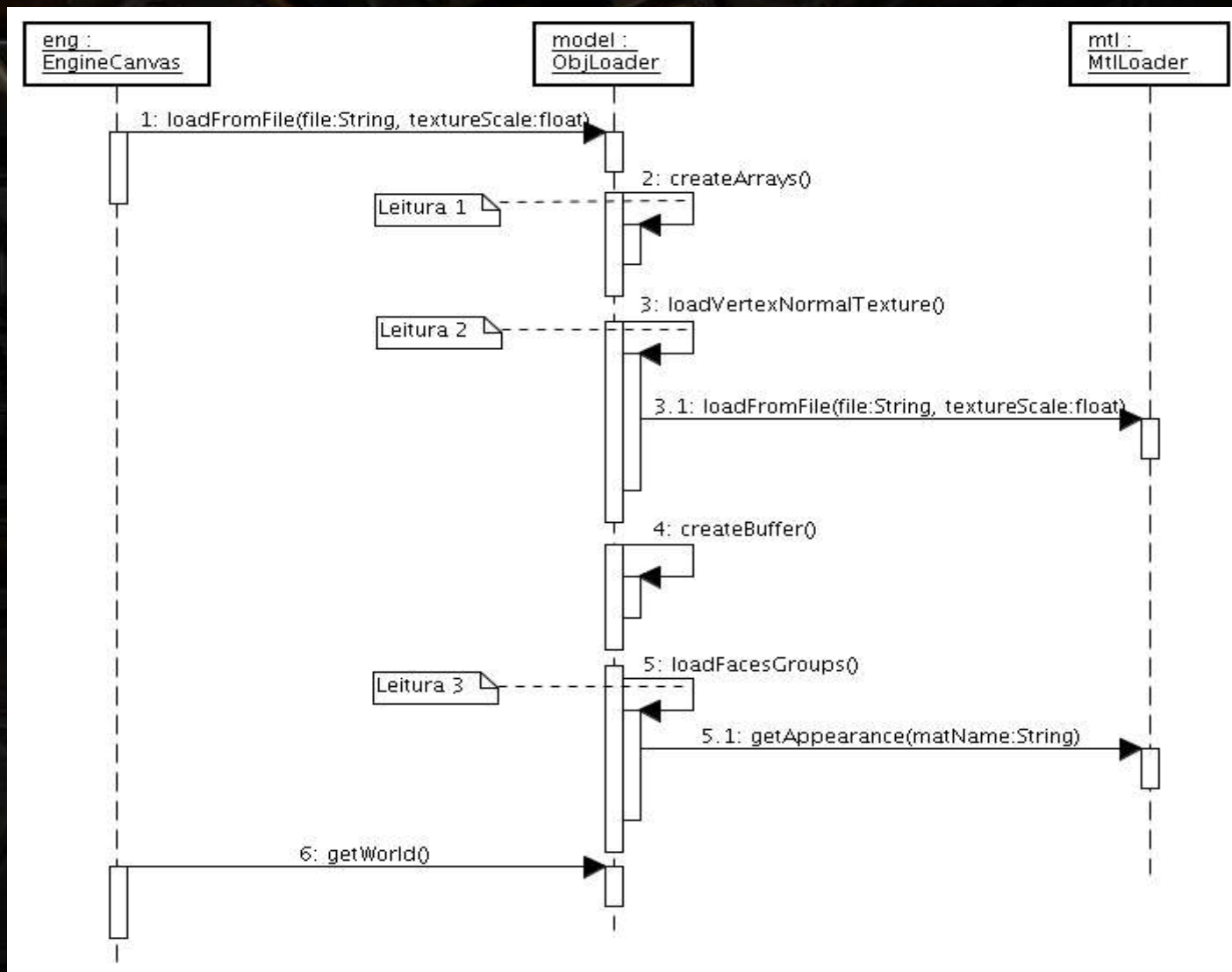
# Mobile 3D Game Engine (M3GE)

- Requisitos:
  - Carregar e desenhar um ambiente virtual a partir de um arquivo de modelo 3D Wavefront
  - Troca de câmeras no cenário
  - Movimentação de personagem no cenário
  - Detecção de colisão básica
  - Modelo de eventos
  - Portabilidade
  - Velocidade

# Arquitetura da Solução



# Carga de Wavefront (Obj e Mtl)





# O Arquivo de Configurações

- Mantém as informações necessárias para que o motor de jogos possa estabelecer seu estágio inicial e o modo de iteração com o usuário
- Define como o jogador irá interagir com o jogo e com que velocidade.
- Define cameras com posições e perspectivas
- Ativa/desativa o cálculo de colisão
- Mantém um valor de dimensionamento das imagens

# Exemplo Arquivo de Configurações

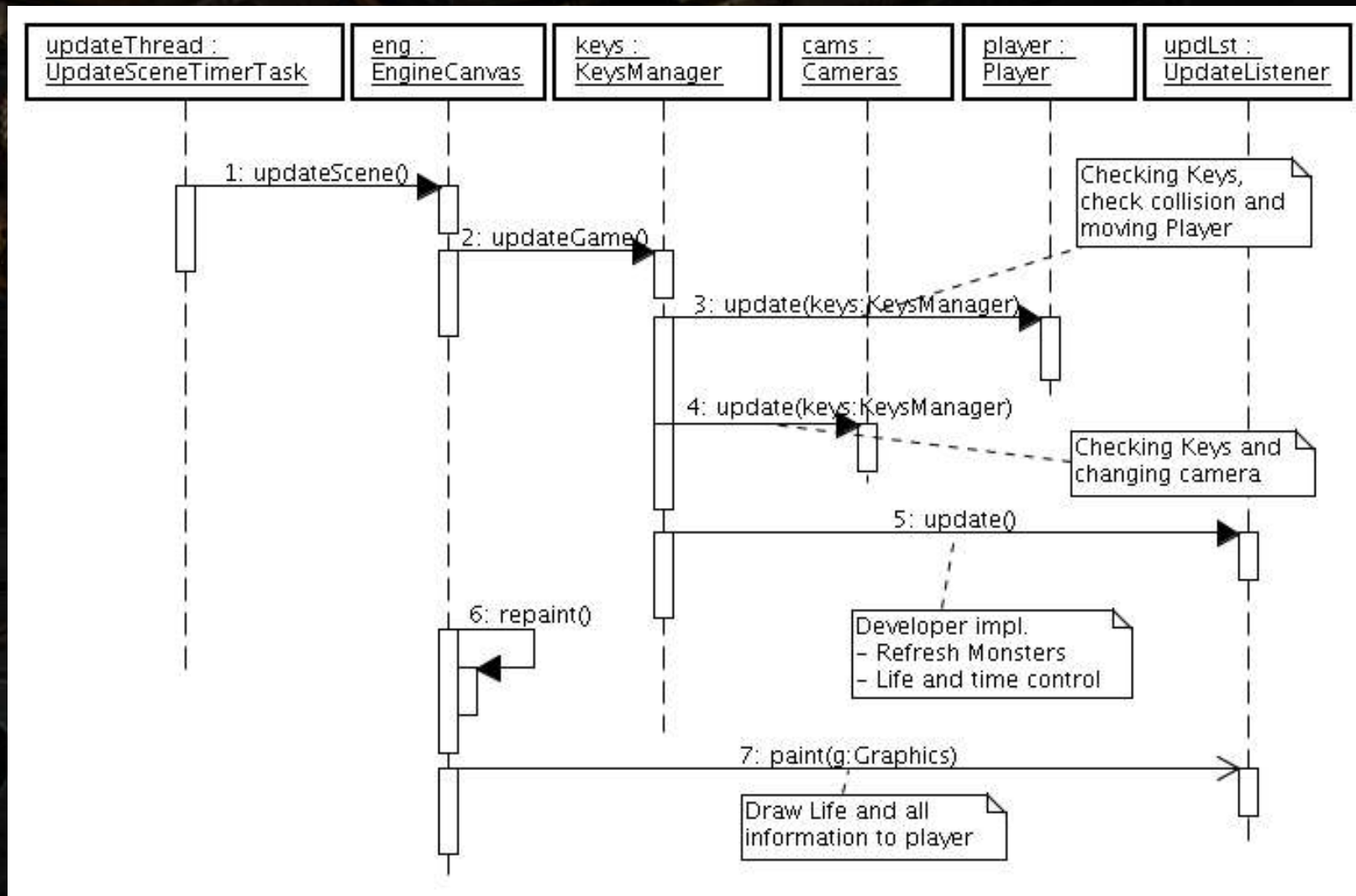
PLAYER\_FRONT= -59  
PLAYER\_BACK= -60  
PLAYER\_LEFT= 55  
PLAYER\_RIGHT= 56  
PLAYER\_JUMP= 51  
PLAYER\_DOWN= 54

VIEW\_UP= 49  
VIEW\_DOWN= 52  
VIEW\_LEFT= -61  
VIEW\_RIGHT= -62  
FIRE=-26

REFRESH\_RATE=50  
TEXTURE\_SCALE=-40  
VELOCITY=0.05  
JUMP\_POWER=0.05  
COLISION\_DETECTION=1  
COLISION\_PLAYER\_RAY=0.05

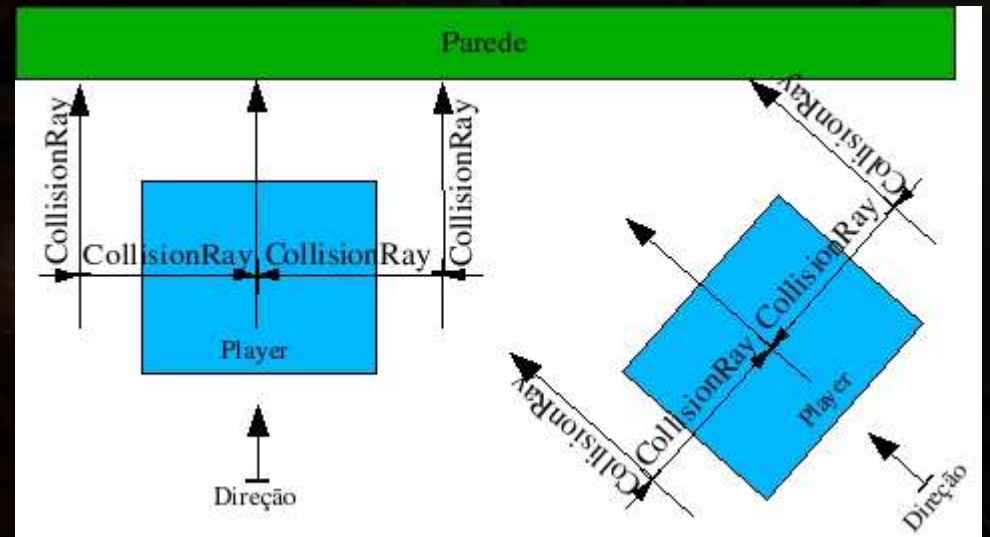
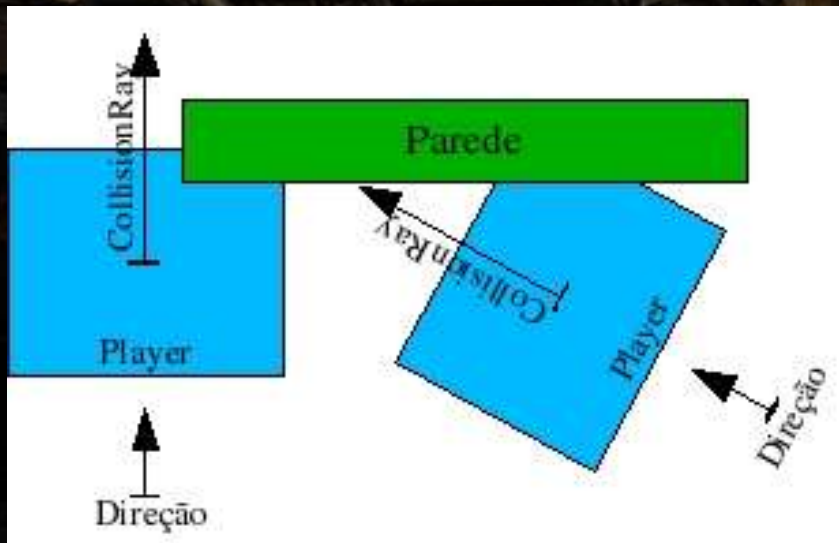
CHANGE\_CAMERA=57  
CAMERA0\_RELATIVE\_TO=PLAYER  
CAMERA0\_X=0  
CAMERA0\_Y=0.00  
CAMERA0\_Z=0  
CAMERA0\_FOVY=70.0  
CAMERA0\_NEAR=0.01  
CAMERA0\_FAR=50.0

# Ciclo Principal do Jogo





# Detecção de colisão



# O Jogo Teste

- Simulação de jogo em primeira pessoa
- Movimentação total pelo cenário
- Detecção de colisão
- 3 Câmeras: olho do personagem, visão externa do personagem e mapa
- Tratamento de frames por segundo, acertos dos tiros, e avisos na troca de câmera.

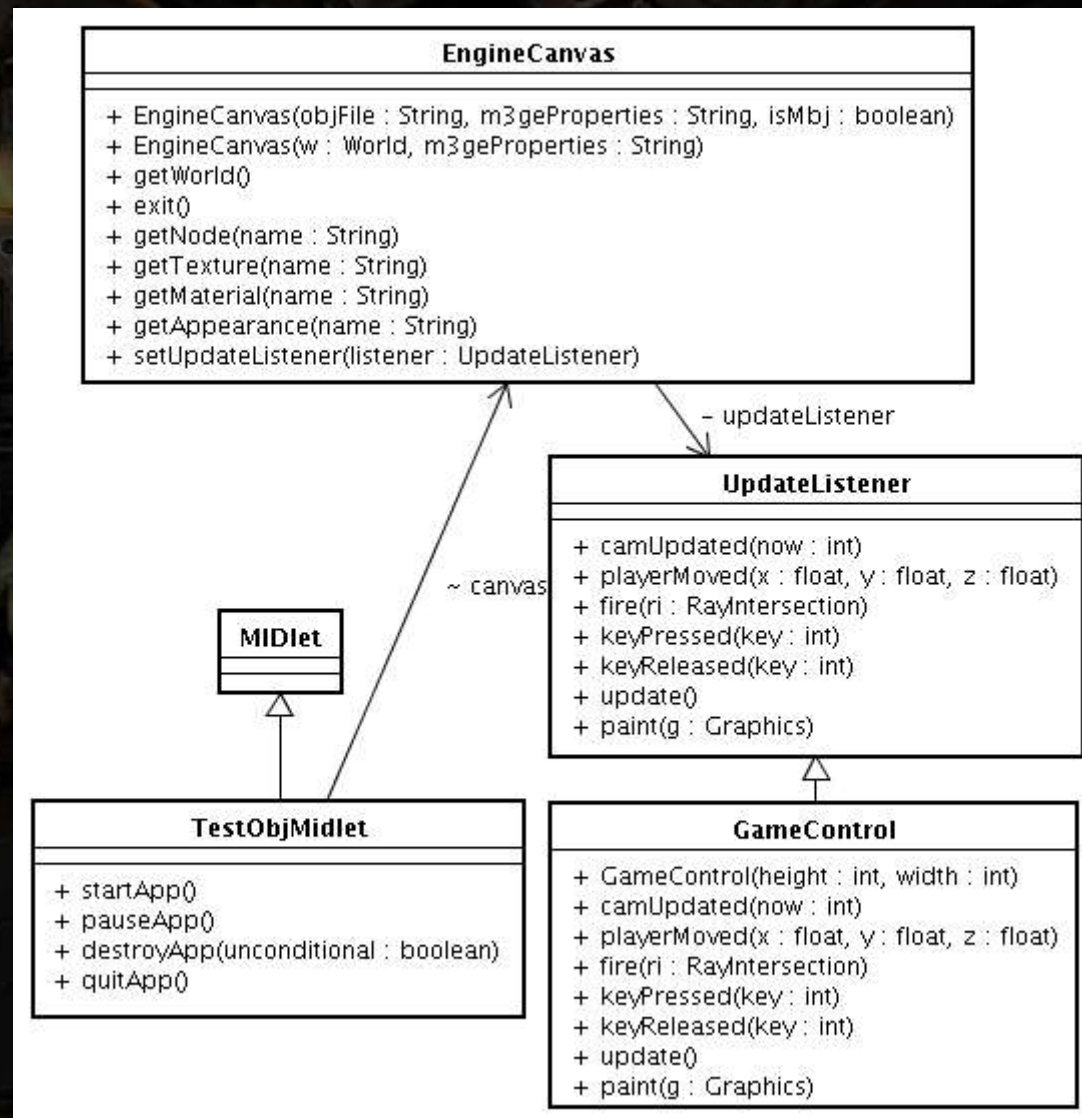


# Jogo Teste





# Diagrama de Classes do Jogo de Testes



# Arquivo MBJ

- Devido a lentidão encontrada ao importar arquivos Obj, definiu-se um novo formato de arquivo.
- O arquivo Obj passa por um pré-processamento em ambiente de desenvolvimento gerando um arquivo Mbj
- Elimina-se o processamento de:
  - Cálculos entre os vetores na importação
  - 3 leituras do arquivo Obj
  - Informações desnecessárias

# Especificação do arquivo MBJ

nf <int>  
nv <int>  
nvt <int>  
nvn <int>  
mtllib <nome do arquivo>  
v <byte> <byte> <byte> <byte> <byte> ....  
vt <short> <short> <short> <short> ...  
vn <byte> <byte> <byte> <byte> <byte> ...

g <float> <float> <float> <nome>  
usemtl <nome do arquivo>  
f <int> <int> <int>  
f <int> <int> <int>  
...



# Resultados

- A única diferença entre emulador e celular é a velocidade de execução.
- Tempo de importação do modelo 3D:
  - Com o arquivo Obj: 17 segundos
  - Com o arquivo Mbj: 14,5 segundos
  - Com a redução do tamanho das texturas: 8 segundos
- Tempo para o JIT: 3 segundos
- O cálculo de colisão leva cerca de 70 milisegundos

# Conclusão

- É possível construir jogos 3D em java para os celulares atuais.
- Os aparelhos estão prontos para os jogos 3D, faltando apenas tornarem-se populares
- Para ganhos com velocidade é necessário abandonar algumas “leis” de engenharia de software, como o uso de recursos avançados da Orientação a Objeto
- Todos os requisitos foram implementados com **SUCESSO**



Obrigado



MOBILEBURN.COM

Vitor Fernando Pamplona