



# Desenvolvimento de Games 2D



1	0	0	1	0	1	0	1	1	1	0	1	0	1	0	0	1	1
1	0	0	0	1	0	1	0	0	1	0	1	0	1	1	0	1	0
1	0	0	0	1	0	0	1	1	1	0	1	1	1	0	1	0	1



# **Desenvolvimento de Games 2D**

**Gabriel Felippe**

**akiradev.netlify.app**

# Introdução

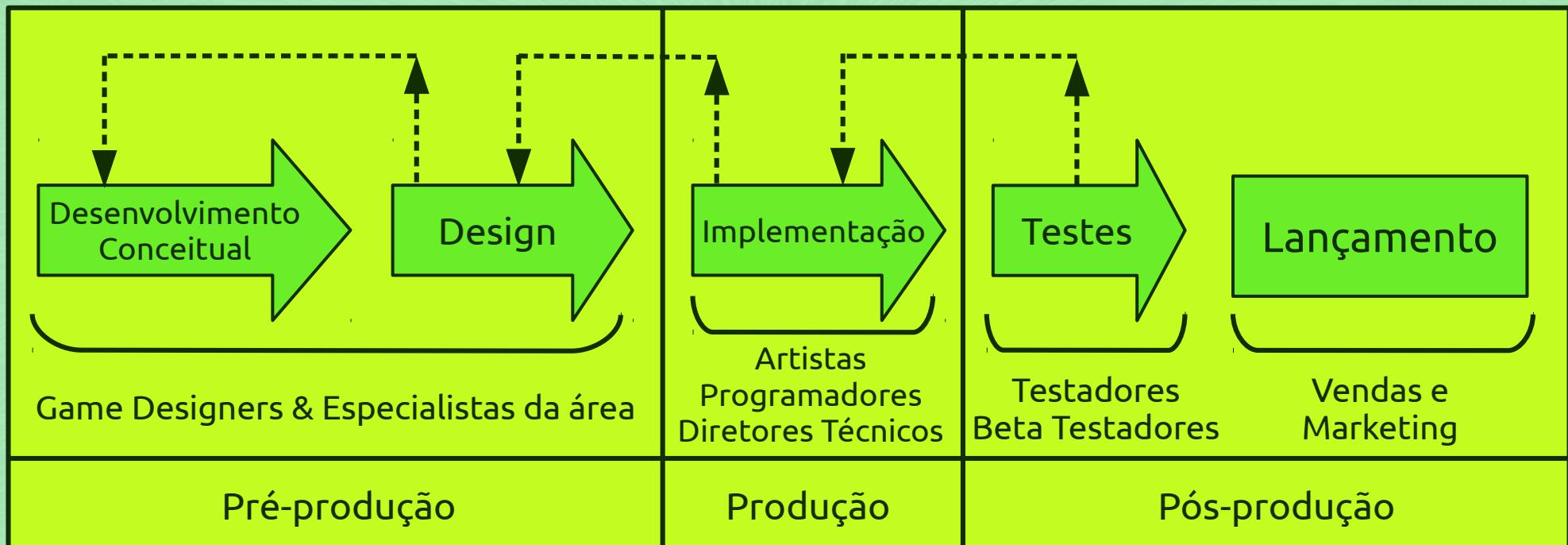
- O desenvolvimento de games é o processo de desenvolvimento de um videogame.
- O esforço é realizado por um desenvolvedor, que varia de uma única pessoa a uma equipe internacional espalhada pelo mundo.
- O desenvolvimento de games comerciais para PC e consoles tradicionais normalmente é financiado por um **publisher** e pode levar vários anos para ser concluído.
- Os games independentes (**Indie Games**) geralmente levam menos tempo e custam menos e podem ser produzidos por indivíduos ou desenvolvedores menores.
- A indústria de games independentes está em grande ascensão, facilitada pelo crescimento de softwares de desenvolvimento de games acessíveis, como a plataforma **Unity**, **Unreal Engine**, **PyGame**, entre outros.

# Introdução

- Os primeiros videogames, desenvolvidos na década de 1960, não eram normalmente comercializados.
- Eles exigiam computadores mainframe para funcionar e não estavam disponíveis para o público em geral.
- O desenvolvimento de games comerciais começou nos anos 70 com o advento dos consoles de videogame de primeira geração e dos primeiros computadores domésticos como o Apple I.
- Os games mainstream comerciais de PC e console são geralmente desenvolvidos em fases: primeiro, na pré-produção, há promoção de venda, os protótipos e os documentos de design do game são escritos; se a ideia for aprovada e o desenvolvedor receber financiamento, o desenvolvimento em grande escala será iniciado.

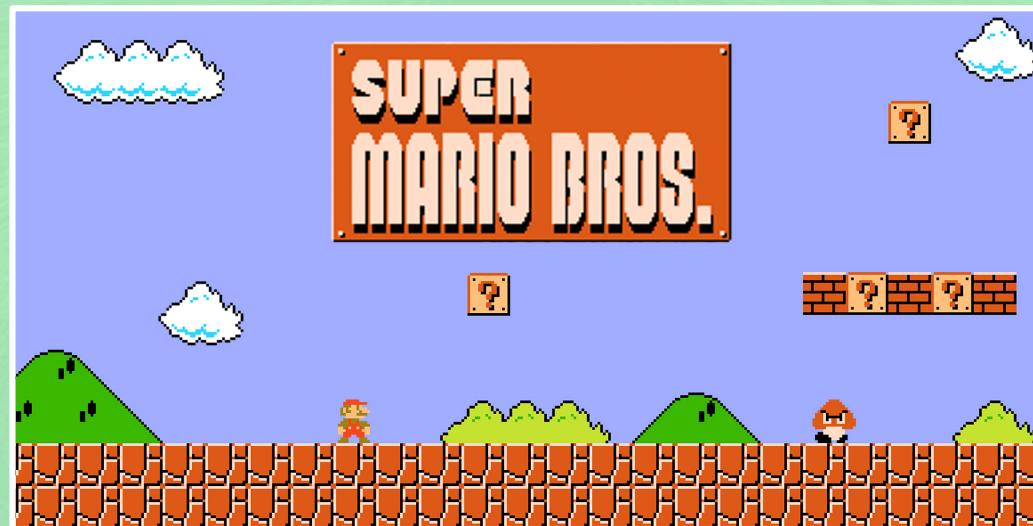
# Introdução

- O desenvolvimento de um game completo geralmente envolve uma equipe de 20 a 100 pessoas com várias responsabilidades, incluindo designers, artistas, programadores e testadores.



# Visão Geral

- Os games são produzidos por meio do processo de desenvolvimento de software.
- Os games são desenvolvidos como uma saída criativa e para gerar lucro.
- A criação de games é considerada arte e ciência.
- O desenvolvimento normalmente é financiado por um **publisher**.

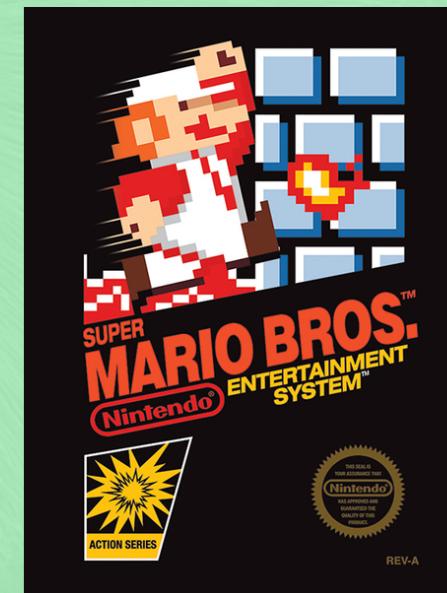


# Visão Geral

- O desenvolvimento de software é o processo de concepção, especificação, design, programação, documentação, teste e correção de bugs envolvidos na criação e manutenção de aplicativos, frameworks, games ou outros componentes de software.
- O desenvolvimento de software é um processo de escrita e manutenção do código-fonte, mas em um sentido mais amplo, inclui tudo o que está envolvido desde a concepção do software desejado até a manifestação final do software, muitas vezes em um processo planejado e estruturado.
- Portanto, o desenvolvimento de software pode incluir pesquisa, novo desenvolvimento, prototipagem, modificação, reutilização, reengenharia, manutenção ou quaisquer outras atividades que resultem em produtos de software.
- Existem diversas abordagens para o gerenciamento de projetos de software, conhecidas como modelos de ciclo de vida de desenvolvimento de software.

# Visão Geral

- Um processo de desenvolvimento de software (também conhecido como metodologia, modelo ou ciclo de vida de desenvolvimento de software) é um framework usado para estruturar, planejar e controlar o processo de desenvolvimento de sistemas de informação.
- A maioria das metodologias compartilha alguma combinação dos seguintes estágios de desenvolvimento de software:
  - ◆ Analisar o problema
  - ◆ Pesquisa de mercado
  - ◆ Coleta de requisitos para o software proposto
  - ◆ Elaborar um plano ou projeto para o software
  - ◆ Implementação (codificação) do software
  - ◆ Teste e debugging do software
  - ◆ Deployment
  - ◆ Manutenção e correção de bugs



# Visão Geral

- Na era inicial dos computadores domésticos e dos consoles de videogame no início dos anos 1980, um único programador podia lidar com quase todas as tarefas de desenvolvimento de um game: programação, design gráfico, efeitos sonoros, etc.
- O desenvolvimento de um game pode demorar até seis semanas. No entanto, as altas expectativas dos usuários e os requisitos dos games comerciais modernos excedem em muito as capacidades de um único desenvolvedor e exigem a divisão de responsabilidades.
- Uma equipe de mais de cem pessoas pode ser empregada em tempo integral para um único projeto.

# Visão Geral

- O desenvolvimento, produção ou design de games é um processo que começa a partir de uma ideia ou conceito.
- Freqüentemente, a ideia é baseada na modificação de um conceito de game existente. A ideia do game pode se enquadrar em um ou vários gêneros.
- Os designers costumam fazer experiências com diferentes combinações de gêneros. Um designer de game geralmente escreve um documento inicial de proposta de game, que descreve o conceito básico, jogabilidade, lista de recursos e características, cenário e história, público-alvo, requisitos e cronograma e, finalmente, estimativas de equipe e orçamento.

# Visão Geral

- Um desenvolvedor de games pode variar de um único indivíduo a uma grande empresa multinacional. Existem estúdios independentes e de propriedade de publishers.
- Desenvolvedores independentes podem contar com suporte financeiro de um publisher de games.
- Normalmente, os desenvolvedores precisam desenvolver um game do conceito ao protótipo, sem financiamento externo. A proposta formal do game é então submetida aos publishers, que podem financiar o desenvolvimento do game por vários meses a anos. O publisher reteria direitos exclusivos para distribuir e comercializar o game e, muitas vezes, deteria os direitos de propriedade intelectual da franquia do game.
- As empresas publishers também pode ser proprietária da empresa do desenvolvedor ou pode ter estúdio(s) de desenvolvimento interno.

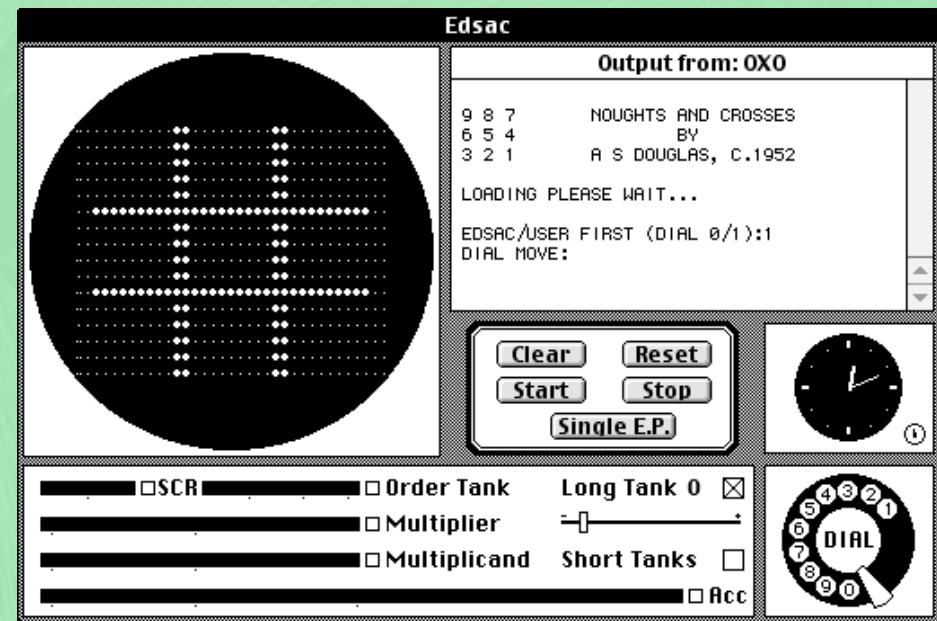
# Visão Geral

- A maioria dos games modernos para PC ou console leva de três a cinco anos para ser concluída, sendo que um game para celular pode ser desenvolvido em poucos meses.
- A duração do desenvolvimento é influenciada por uma série de fatores, como gênero, escala, plataforma de desenvolvimento e número de ativos.
- Alguns games podem demorar muito mais do que o prazo médio para serem concluídos.
- Um exemplo infame é Duke Nukem Forever da 3D Realms, anunciado para estar em produção em abril de 1997 e lançado quatorze anos depois, em junho de 2011.

# História

- A história da criação de games começa com o desenvolvimento dos primeiros videogames, embora qual videogame seja o primeiro depende da definição de videogame.
- Os primeiros games criados tinham pouco valor de entretenimento e seu foco de desenvolvimento era separado da experiência do usuário - na verdade, esses games exigiam computadores mainframe para jogá-los.
- OXO, escrito por Alexander S. Douglas em 1952, foi o primeiro game de computador a usar um display digital.

OXO reproduzido em um simulador EDSAC para o Mac OS clássico



# História

- Em 1958, um game chamado Tennis for Two, que exibia seu resultado em um osciloscópio, foi feito por Willy Higinbotham, um físico que trabalhava no Brookhaven National Laboratory.

Tennis for Two em um osciloscópio DuMont Lab tipo 304-A



# História

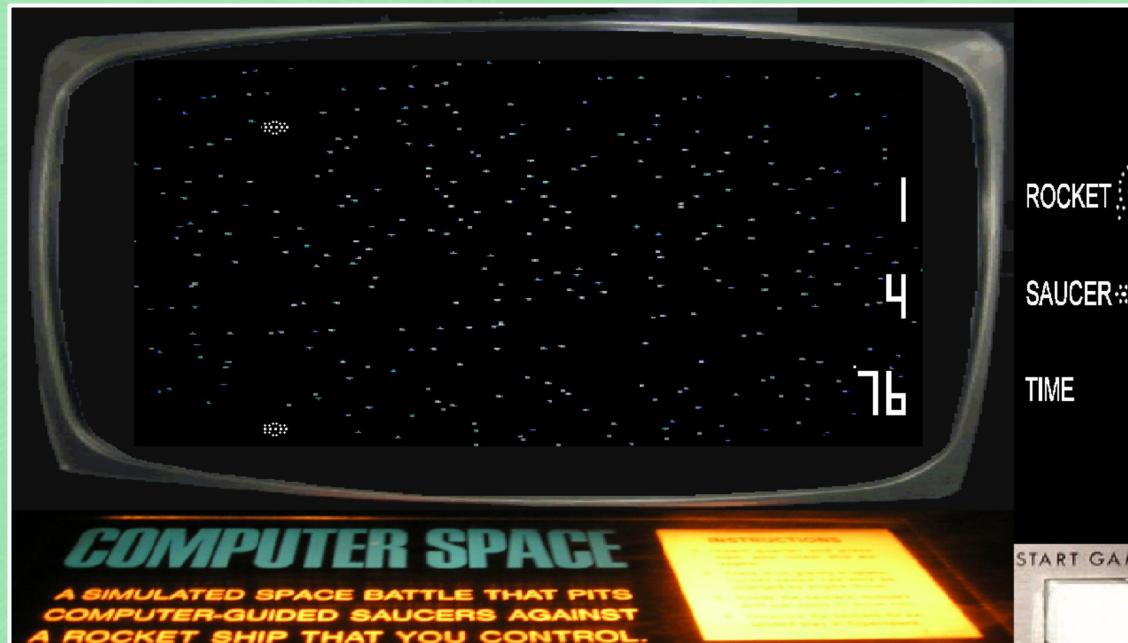
- Em 1961, um game de computador mainframe chamado Spacewar! foi desenvolvido por um grupo de alunos do Massachusetts Institute of Technology liderado por Steve Russell.

Spacewar! no PDP-1 do  
Computer History  
Museum em 2007



# História

- O verdadeiro design comercial e o desenvolvimento de games começaram na década de 1970, quando os videogames de arcade e os consoles de primeira geração foram comercializados. Em 1971, Computer Space foi o primeiro videogame operado por moedas vendido comercialmente. Ele usava uma televisão em preto e branco como tela, e o sistema do computador era feito de 74 chips TTL.



# História

- Em 1972, o primeiro sistema de console doméstico foi lançado, chamado Magnavox Odyssey, desenvolvido por Ralph H. Baer.
- No mesmo ano, a Atari lançou Pong, um game de arcade que aumentou a popularidade dos videogames. O sucesso comercial de Pong levou outras empresas a desenvolver clones de Pong, gerando a indústria de videogames.



# História

- Os programadores trabalharam em grandes empresas para produzir games para esses dispositivos. A indústria não viu uma grande inovação no design de games e um grande número de consoles tinha games muito semelhantes.
- Muitos desses primeiros games eram geralmente clones de Pong. Alguns games eram diferentes, no entanto, teve Gun Fight, que era significativo por vários motivos: um game de tiro multidirecional do início de 1975, que retratava personagens, violência e combate de humanos com humanos.
- A versão original de Tomohiro Nishikado foi baseada em lógica discreta, que Dave Nutting adaptou usando o Intel 8080, tornando ele o primeiro videogame a usar um microprocessador.

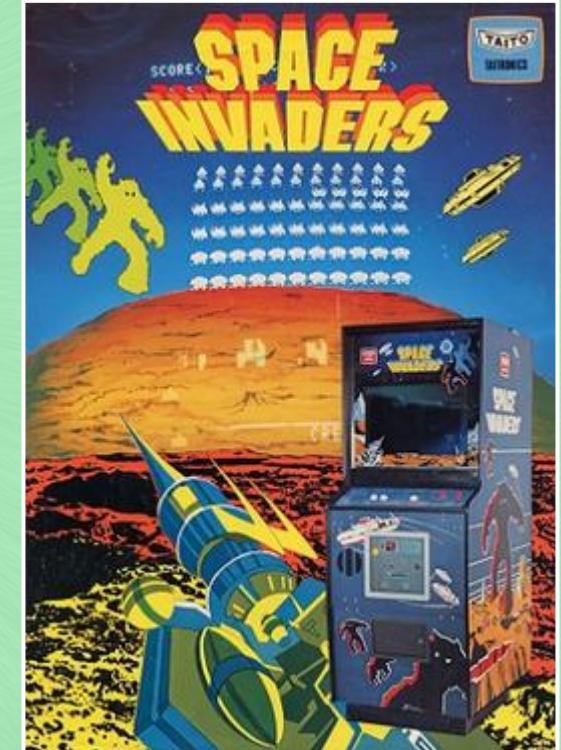
# História

- Os fabricantes de consoles logo começaram a produzir consoles que eram capazes de jogar games desenvolvidos de forma independente e rodavam em microprocessadores, marcando o início dos consoles de segunda geração, começando com o lançamento do Fairchild Channel F em 1976.



# História

- A inundação de clones de Pong levou ao crash do videogame em 1977, que acabou chegando ao fim com o sucesso do game de tiro de arcade da Taito de 1978, Space Invaders, marcando o início da era de ouro dos videogames e inspirando dezenas de fabricantes entrar no mercado.
- Seu criador Nishikado não apenas projetou e programou o game, mas também fez a arte, projetou o hardware de arcade e montou um microcomputador do zero. Ele logo foi portado para o Atari 2600, tornando-se o primeiro "aplicativo matador" e quadruplicando as vendas do console.



# História

- Ao mesmo tempo, os computadores domésticos surgiram no mercado, permitindo que programadores individuais e amadores desenvolvessem games.
- Isso permitiu que os fabricantes de hardware e software agissem separadamente.
- Um número muito grande de games poderia ser produzido por um indivíduo, pois os games eram fáceis de fazer porque a limitação gráfica e de memória não permitia muito conteúdo.
- Desenvolveram-se então empresas maiores, que focaram equipes selecionadas para trabalhar em um título.
- Os desenvolvedores de muitos dos primeiros videogames domésticos, como Zork, Baseball, Air Warrior e Adventure, mais tarde fizeram a transição de seu trabalho como produtos da indústria dos primeiros videogames.

# História

- A indústria se expandiu significativamente na época, com o setor de videogame sozinho (representando a maior parte da indústria de games) gerando receitas maiores do que a música pop e os filmes de Hollywood juntos.
- A indústria de videogames domésticos, no entanto, sofreu grandes perdas após o crash do videogame em 1983.
- Em 1984, Jon Freeman alertou na Computer Gaming World:  
“Are computer games the way to fame and fortune? No. Not unless your idea of fame is having your name recognized by one or two astute individuals at Origins... I've been making a living (after a fashion) designing games for most of the last six years. I wouldn't recommend it for someone with a weak heart or a large appetite, though.”

# História

- Chris Crawford e Don Daglow em 1987, da mesma forma, aconselharam os designers em potencial a escrever games primeiro como um hobby e a não largar seus trabalhos antes do tempo. A indústria de videogames caseiros foi revitalizada logo depois com o amplo sucesso do Nintendo Entertainment System.



# História

- Compute!'s Gazette em 1986 afirmou que embora indivíduos tenham desenvolvido a maioria dos primeiros videogames, "É impossível para uma pessoa ter os múltiplos talentos necessários para criar um bom game". Em 1987, um videogame exigia 12 meses para ser desenvolvido e outros seis para planejar o marketing. Os projetos geralmente eram esforços individuais, com desenvolvedores sozinhos entregando games acabados para seus editores. Com o processamento e as capacidades gráficas cada vez maiores de produtos de arcade, console e computador, junto com um aumento nas expectativas do usuário, o design de game foi além do escopo de um único desenvolvedor para produzir um game comercializável. The Gazette declarou: "O processo de escrever um game envolve chegar a um conceito original e divertido, ter a habilidade de concretizá-lo por meio de uma programação boa e eficiente, e também ser um artista bastante respeitável". Isso desencadeou o início do desenvolvimento baseado em equipe. Em termos gerais, durante a década de 1980, a pré-produção envolvia esboços e rotinas de teste de único desenvolvedor. Na década de 1990, a pré-produção consistia principalmente em visualizações de game art. No início dos anos 2000, a pré-produção geralmente produzia uma demo jogável.

# História

- Em 2000, um projeto de desenvolvimento de 12 a 36 meses foi financiado por uma editora por US\$1M–3M. Além disso, US\$250 mil a 1.5 milhão foram gastos em marketing e desenvolvimento de vendas.
- Em 2001, mais de 3.000 games foram lançados para PC; e cerca de 100 games geraram lucro, sendo que apenas cerca de 50 tiveram lucro significativo. No início dos anos 2000, tornou-se cada vez mais comum usar middleware game engines, como Quake engine ou Unreal engine.



# História

- No início dos anos 2000, os mobile games também começaram a ganhar popularidade. No entanto, os mobile games distribuídos por operadoras de celular permaneceram uma forma marginal de game até o lançamento da App Store da Apple em 2008.

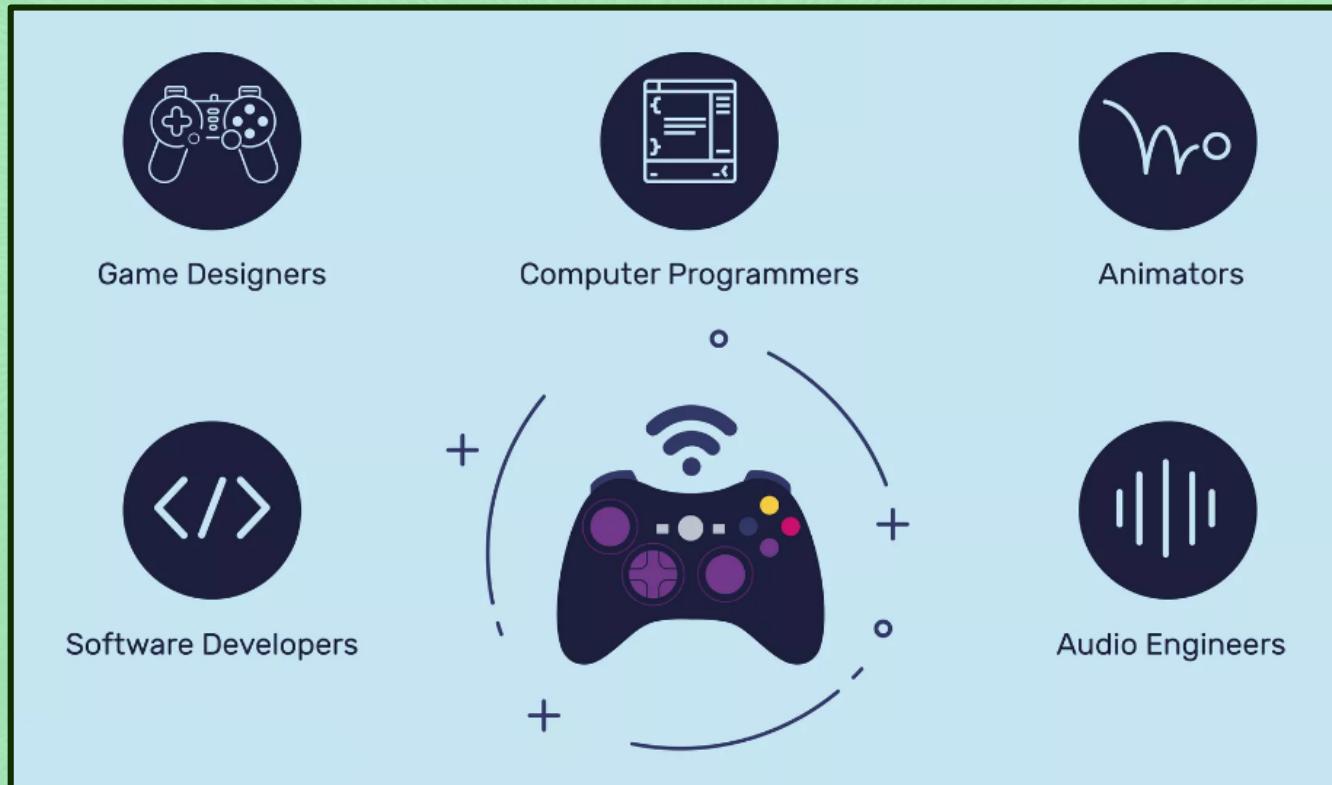


# História

- A Apple App Store, lançada em 2008, foi a primeira loja de aplicativos móveis operada diretamente pelo titular da plataforma móvel. Ela mudou significativamente o comportamento do consumidor, mais favorável para download de conteúdo móvel e rapidamente ampliou os mercados de games móveis.
- Em 2009, o valor anual de mercado dos games foi estimado entre US\$7–30 bilhões, dependendo de quais números de vendas estão incluídos. Isso está no mesmo nível do mercado de bilheteria de filmes. Um publisher normalmente financiaria um desenvolvedor independente por \$500k–\$5 milhões para o desenvolvimento de um título.
- Em 2012, o valor total anual já havia chegado a US\$66.3 bilhões e, àquela altura, os mercados de videogame não eram mais dominados por videogames. De acordo com Newzoo, a participação de MMO's foi de 19.8%, PC/MAC's 9.8%, tablets 3.2%, smartphones 10.6%, handhelds 9.8%, consoles apenas 36.7% e games online casuais 10.2%. Os segmentos de mercado que mais crescem são os games para celular, com uma taxa média anual de 19% para smartphones e 48% para tablets.

# Funções

- Existem diversas funções na indústria de games, a seguir veremos algumas das mais conhecidas.



# Produtor

- O desenvolvimento é supervisionado por produtores internos e externos.
- O produtor que trabalha para o desenvolvedor é conhecido como o produtor interno e gerencia a equipe de desenvolvimento, programa, relata o progresso, contrata e designa a equipe e assim por diante.
- O produtor que trabalha para o publisher é conhecido como produtor externo e supervisiona o progresso e o orçamento do desenvolvedor.
- As responsabilidades do produtor incluem relações públicas, negociação de contrato, contato entre a equipe e as partes interessadas, manutenção de cronograma e orçamento, garantia de qualidade, gerenciamento de teste beta e localização.
- Essa função também pode ser referida como gerente de projeto, líder de projeto ou diretor.

# Publisher

- Um publisher de videogame é uma empresa que publica videogames que desenvolveu internamente ou foi desenvolvido por um desenvolvedor de videogame externo.
- Assim como acontece com as editoras de livros ou de filmes em DVD, as publishers de videogames são responsáveis pela fabricação e pelo marketing de seus produtos, incluindo pesquisa de mercado e todos os aspectos da publicidade.
- Eles geralmente financiam o desenvolvimento, às vezes pagando um desenvolvedor de videogame (o publisher chama isso de desenvolvimento externo) e às vezes pagando uma equipe interna de desenvolvedores chamada de estúdio.
- Outras funções normalmente desempenhadas pelo publisher incluem decidir e pagar por qualquer licença que o game possa utilizar; layout, impressão e possivelmente a redação do manual do usuário; e a criação de elementos de design gráfico.

# Time de Desenvolvimento

- Os desenvolvedores podem variar em tamanho, desde pequenos grupos criando games casuais até abrigar centenas de funcionários e produzir vários títulos grandes.
- As empresas dividem suas subtarefas de desenvolvimento de games.
- Os cargos individuais podem variar; no entanto, as funções são as mesmas dentro da indústria. A equipe de desenvolvimento é composta por vários membros.
- Alguns membros da equipe podem desempenhar mais de uma função; da mesma forma, mais de uma tarefa pode ser realizada pelo mesmo membro.
- O tamanho da equipe pode variar de 20 a 100 ou mais membros, dependendo do escopo do game.
- Os mais representados são os artistas, seguidos pelos programadores, depois designers e, por fim, especialistas em áudio, com dois a três produtores na gestão.

# Designer

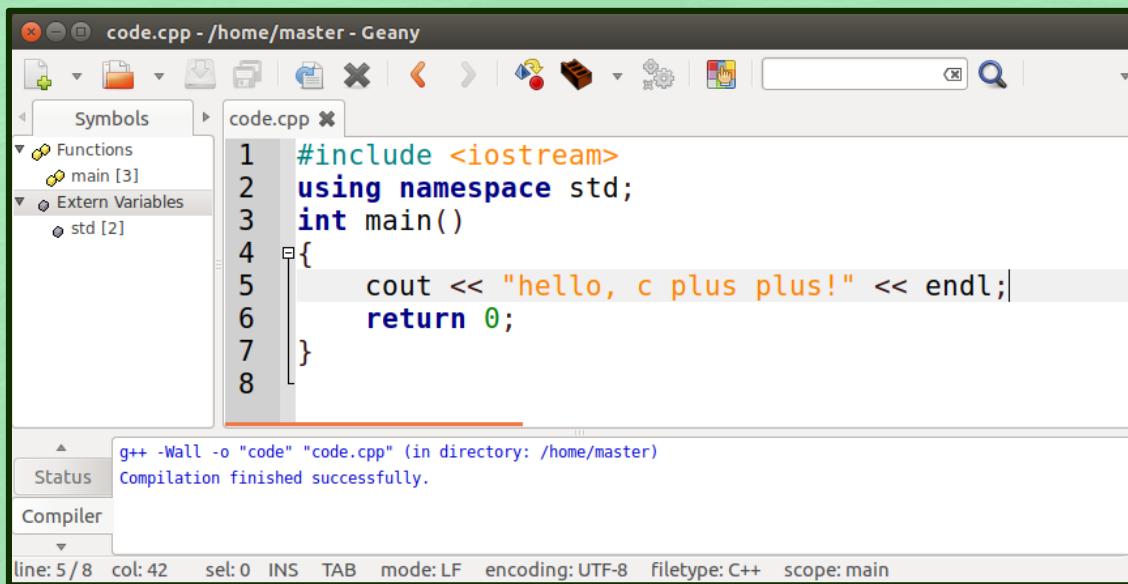
- Um designer de games é uma pessoa que projeta a jogabilidade, concebendo e projetando as regras e a estrutura de um game.
- As equipes de desenvolvimento geralmente têm um designer-chefe que coordena o trabalho de outros designers. Eles são os principais visionários do game. Uma das funções de um designer é ser um escritor, para conceber a narrativa, o diálogo, o comentário, a narrativa cutscene do game, os diários, o conteúdo da embalagem do videogame, o sistema de dicas, etc.
- Em projetos maiores, geralmente há designers separados para várias partes do game, como mecânica do game, interface do usuário, personagens, diálogos, gráficos, etc.

# Artista

- Um artista de jogos é um artista visual que cria arte de videogame.
- A produção de arte geralmente é supervisionada por um diretor de arte ou líder de arte, garantindo que sua visão seja seguida.
- O diretor de arte gerencia a equipe de arte, agendando e coordenando dentro da equipe de desenvolvimento.
- O trabalho do artista pode ser orientado para 2D ou 3D.
- Artistas 2D podem produzir arte conceitual, sprites, texturas, cenários ambientais ou imagens de terreno e interface de usuário.
- Os artistas 3D podem produzir modelos ou meshes, animação, ambiente 3D e cinematográfica.
- Os artistas às vezes ocupam os dois papéis.

# Programador

- Um programador de games é um engenheiro de software que desenvolve principalmente videogames ou software relacionado (como ferramentas de desenvolvimento de games).
- O desenvolvimento da base de código do game é feito por programadores.
- Geralmente há um ou vários programadores líderes, que implementam a base de código inicial do game e fazem uma visão geral do desenvolvimento futuro e da alocação do programador em módulos individuais.



The screenshot shows the Geany integrated development environment (IDE) interface. The main window displays a code editor with the file 'code.cpp' open. The code content is:

```
#include <iostream>
using namespace std;
int main()
{
    cout << "hello, c plus plus!" << endl;
    return 0;
}
```

The status bar at the bottom provides compiler information: 'g++ -Wall -o "code" "code.cpp" (in directory: /home/master)' and 'Compilation finished successfully.' The status bar also shows the current line (line 5), column (col 42), and other editing modes.

# Programador

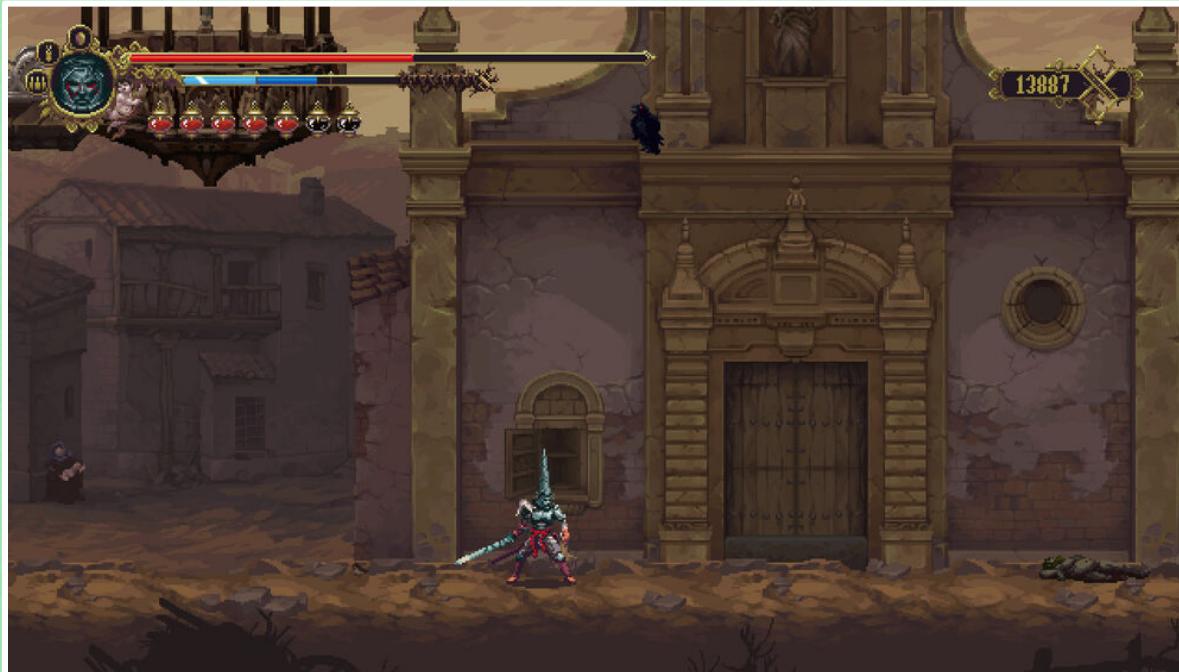
- As funções individuais das disciplinas de programação incluem:
  - ◆ **Física:** a programação do game engine, incluindo simulação de física, colisão, movimento de objetos, etc .
  - ◆ **Inteligência Artificial:** produzindo agentes de computador usando técnicas de IA de game, como scripts, planejamento, decisões baseadas em regras, etc.
  - ◆ **Gráficos:** o gerenciamento de utilização de conteúdo gráfico e considerações de memória; a produção de motor gráfico, integração de modelos, texturas para trabalhar junto ao motor de física.
  - ◆ **Som:** integração de música, fala, efeitos sonoros nos locais e horários adequados.
  - ◆ **Jogabilidade:** implementação de várias regras e recursos do game (às vezes chamado de generalista).

# Programador

- As funções individuais das disciplinas de programação incluem:
  - ◆ **Scripting:** desenvolvimento e manutenção de sistema de comando de alto nível para várias tarefas no game, como IA, level editor triggers, etc.
  - ◆ **Interface de Usuário:** produção de elementos de interface do usuário, como menus de opções, HUDs, sistemas de ajuda e feedback, etc.
  - ◆ **Processamento de entrada:** correlação de processamento e compatibilidade de vários dispositivos de entrada, como teclado, mouse, gamepad, etc.
  - ◆ **Comunicações de rede:** o gerenciamento de entradas e saídas de dados para game locais e na Internet.
  - ◆ **Ferramentas de game:** produção de ferramentas para acompanhar o desenvolvimento do game, especialmente para designers e roteiristas.

# Level Designer

- Um designer de níveis é uma pessoa que cria níveis, desafios ou missões para videogames usando um conjunto específico de programas. Esses programas podem ser programas comerciais de design 3D ou 2D comumente disponíveis, ou editores de nível especialmente projetados e personalizados feitos para um game específico.

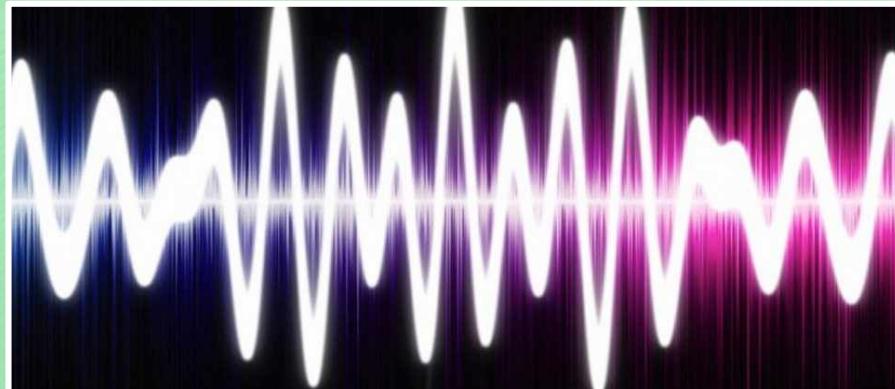


# Level Designer

- Os designers de níveis trabalham com versões incompletas e completas do game.
- Os programadores de games geralmente produzem editores de níveis e ferramentas de design para os designers usarem.
- Isso elimina a necessidade dos designers de acessar ou modificar o código do jogo.
- Os editores de nível podem envolver linguagens de script de alto nível personalizadas para ambientes interativos ou IAs.
- Ao contrário das ferramentas de edição de níveis às vezes disponíveis para a comunidade, os designers de níveis geralmente trabalham com espaços reservados e protótipos visando à consistência e layout claro antes que a arte final necessária seja concluída.

# Engenheiro de Som

- Os engenheiros de som são profissionais técnicos responsáveis pelos efeitos sonoros e posicionamento sonoro.
- Eles às vezes supervisionam a dublagem e a criação de outros recursos de som.
- Os compositores que criam a trilha sonora de um game também compõem a equipe de som do game, embora muitas vezes esse trabalho seja terceirizado.

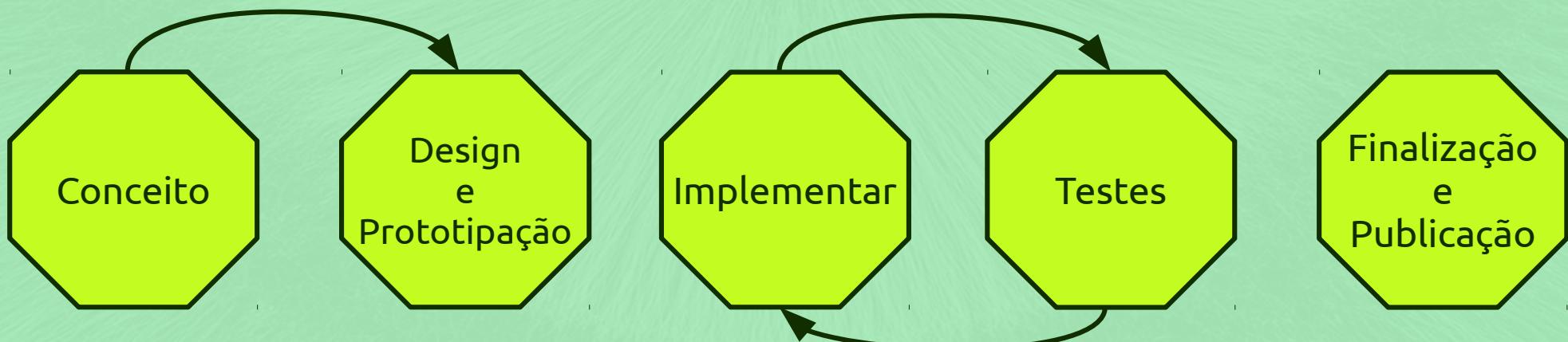


# Testador

- A garantia de qualidade é realizada por testadores de games.
- Um testador de games analisa videogames para documentar defeitos de software como parte de um controle de qualidade.
- O teste é um campo altamente técnico que requer experiência em computação e competência analítica.
- Os testadores garantem que o game se enquadre no design proposto: funciona e é divertido.
- Isso envolve o teste de todos os recursos, compatibilidade, etc. Embora seja necessário em todo o processo de desenvolvimento, o teste é caro e muitas vezes utilizado ativamente apenas para a conclusão do projeto.

# Processo de Desenvolvimento

- O desenvolvimento de games é um processo de desenvolvimento de software, uma vez que um videogame é um software com arte, áudio e jogabilidade.
- Os métodos formais de desenvolvimento de software são freqüentemente esquecidos.
- Games com metodologia de desenvolvimento pobre provavelmente ultrapassarão o orçamento e as estimativas de tempo, além de conter um grande número de bugs. O planejamento é importante para projetos individuais e em grupo.



# Processo de Desenvolvimento

- O desenvolvimento geral de games não é adequado para métodos típicos de ciclo de vida de software, como o modelo em cascata.
- Um método empregado para o desenvolvimento de games é o desenvolvimento ágil. Ele é baseado na prototipagem iterativa, um subconjunto da prototipagem de software.
- O desenvolvimento ágil depende do feedback e do refinamento das iterações do game com o conjunto de features aumentando gradualmente. Este método é eficaz porque a maioria dos projetos não começa com um esboço claro de requisitos. Um método popular de desenvolvimento ágil de software é o Scrum.
- A seguir veremos alguns estágios do desenvolvimento de games comerciais.

# Pré-produção

- A fase de pré-produção ou design é uma fase de planejamento do projeto com foco no desenvolvimento de ideias e conceitos e na produção dos documentos iniciais de design.
- O objetivo do desenvolvimento de conceito é produzir documentação clara e fácil de entender, que descreva todas as tarefas, cronogramas e estimativas para a equipe de desenvolvimento.
- O conjunto de documentos produzidos nesta fase é denominado plano de produção. Esta fase geralmente não é financiada por um publisher, no entanto, bons publishers podem exigir que os desenvolvedores produzam planos durante a pré-produção.
- O estágio final da pré-produção também pode ser referido como prova de conceito ou revisão técnica quando documentos mais detalhados do game são produzidos.

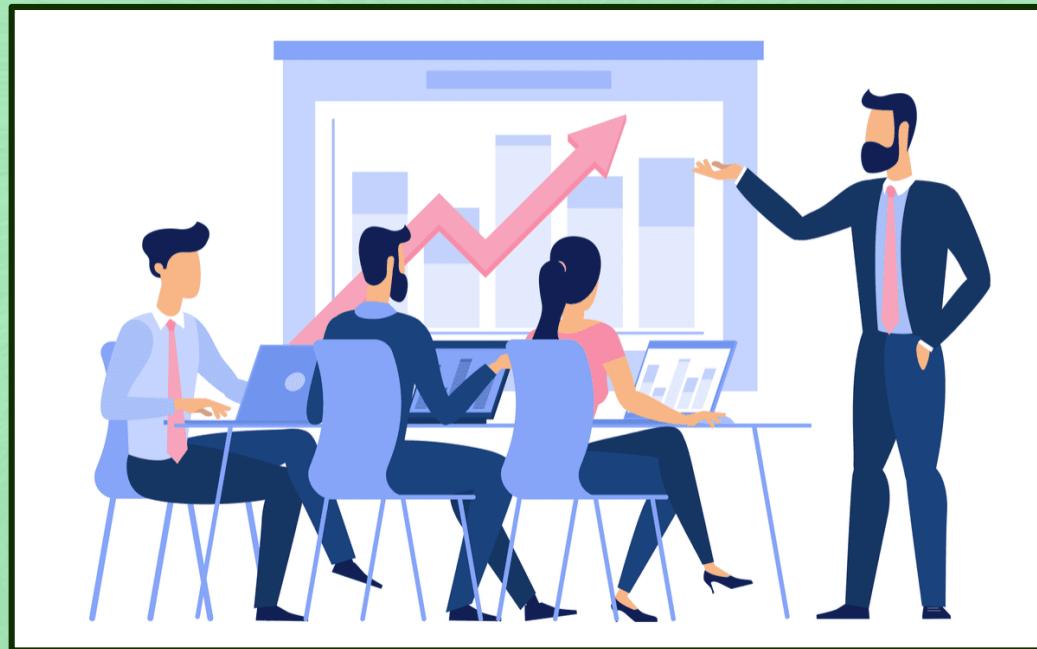
# Alto Conceito

- O alto conceito é uma breve descrição de um game. O alto conceito é a resposta de uma ou duas frases à pergunta: "Sobre o que é o seu game?".



# Pitch

- Um pitch (argumento de venda), documento de conceito, documento de proposta ou proposta de game é um pequeno documento de resumo destinado a apresentar os argumentos de venda do game e detalhar porque o game seria lucrativo para desenvolver.



# Conceito

- Documento de conceito, proposta de game ou plano de game é um documento mais detalhado do que o documento de argumento de venda.
- Isso inclui todas as informações produzidas sobre o game. Incluindo alto conceito, gênero do game, descrição da jogabilidade, recursos, cenário, história, público-alvo, plataformas de hardware, cronograma estimado, análise de marketing, requisitos da equipe e análise de risco.
- Antes que um projeto aprovado seja concluído, uma equipe mínima de programadores e artistas geralmente começa a trabalhar.
- Os programadores podem desenvolver protótipos rápidos, apresentando um ou mais recursos que as partes interessadas gostariam de ver incorporados no produto final.
- Os artistas podem desenvolver arte conceitual e esboços como um trampolim para o desenvolvimento do game real.

# Game Design Document

- Antes que uma produção em grande escala possa começar, a equipe de desenvolvimento produz a primeira versão de um documento de design de game incorporando todo ou a maior parte do material do passo inicial.
- O documento de design descreve o conceito do game e os principais elementos da jogabilidade em detalhes.
- Também pode incluir esboços preliminares de vários aspectos do game.
- O documento de design às vezes é acompanhado por protótipos funcionais de algumas seções do game.
- O documento de design permanece um documento vivo durante todo o desenvolvimento - frequentemente alterado semanalmente ou mesmo diariamente.
- Compilar uma lista das necessidades do game é chamado de "captura de requisitos".

# Protótipo

- Escrever protótipos de ideias e recursos de jogabilidade é uma atividade importante que permite que programadores e designers de games experimentem diferentes algoritmos e cenários de usabilidade para um game.



# Protótipo

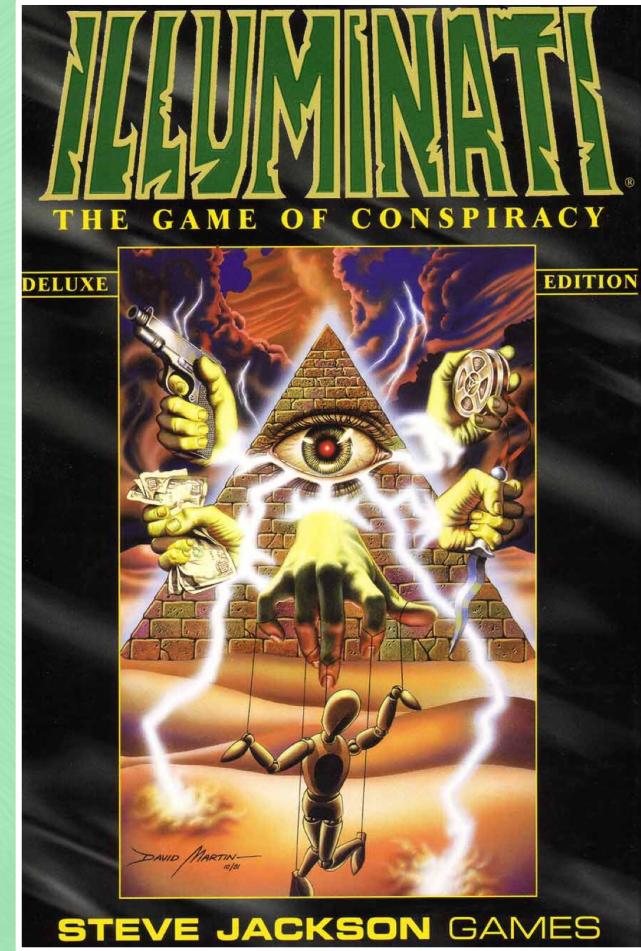
- Uma grande quantidade de prototipagem pode ocorrer durante a pré-produção antes que o documento de design seja concluído e pode, de fato, ajudar a determinar quais recursos o design especifica.
- A prototipagem neste estágio geralmente é feita manualmente (prototipagem em papel), não digitalmente, já que geralmente é mais fácil e rápido testar e fazer alterações antes de perder tempo e recursos no que poderia ser uma ideia ou projeto cancelado.
- A prototipagem também pode ocorrer durante o desenvolvimento ativo para testar novas ideias à medida que o game surge.
- Os protótipos geralmente servem apenas para atuar como uma prova de conceito ou para testar ideias, adicionando, modificando ou removendo alguns dos recursos.
- A maioria dos algoritmos e recursos lançados em um protótipo podem ser transferidos para o game depois de concluídos.

# Produção

- A produção é o estágio principal de desenvolvimento, quando os assets e o código-fonte do game são produzidos.
- Os programadores escrevem um novo código-fonte, os artistas desenvolvem game assets, como sprites ou modelos 3D.
- Os engenheiros de som desenvolvem efeitos sonoros e os compositores desenvolvem música para o game.
- Designers de níveis criam níveis e escritores escrevem diálogos para cenas e NPCs.
- Os designers de games continuam a desenvolver o design do game durante a produção.

# Design

- O design de games é um processo essencial e colaborativo de criação do conteúdo e das regras de um game, exigindo competência artística e técnica, bem como habilidades de redação. Criatividade e uma mente aberta são vitais para a conclusão de um videogame de sucesso.
- Durante o desenvolvimento, o designer do game implementa e modifica o design do game para refletir a visão atual do game. Recursos e níveis são freqüentemente removidos ou adicionados. O tratamento da arte pode evoluir e a história de fundo pode mudar. Uma nova plataforma pode ser direcionada, bem como um novo grupo demográfico.
- Todas essas mudanças precisam ser documentadas e divulgadas para o restante da equipe. A maioria das alterações ocorre como atualizações no documento de design.



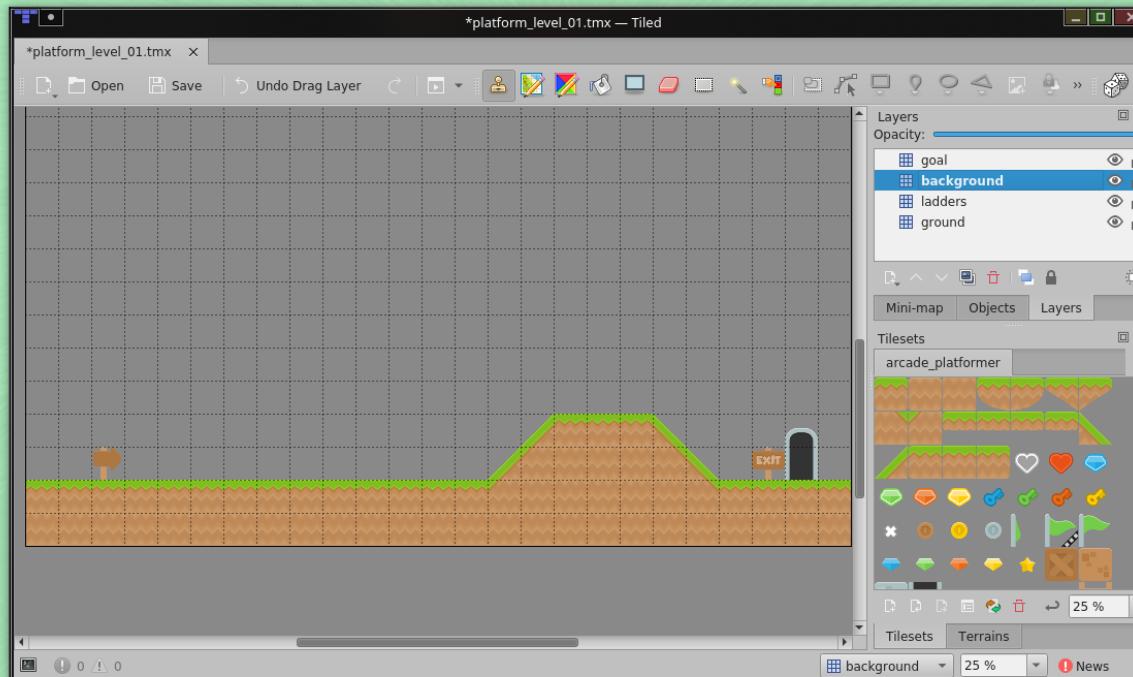
# Programação

- A programação do game é feita por um ou mais programadores de games.
- Eles desenvolvem protótipos para testar ideias, muitas das quais podem nunca chegar ao game final.
- Os programadores incorporam novos recursos exigidos pelo design do game e corrigem quaisquer bugs introduzidos durante o processo de desenvolvimento.
- Mesmo se um game engine pronto para uso for usado, uma grande quantidade de programação é necessária para personalizar quase todos os games.



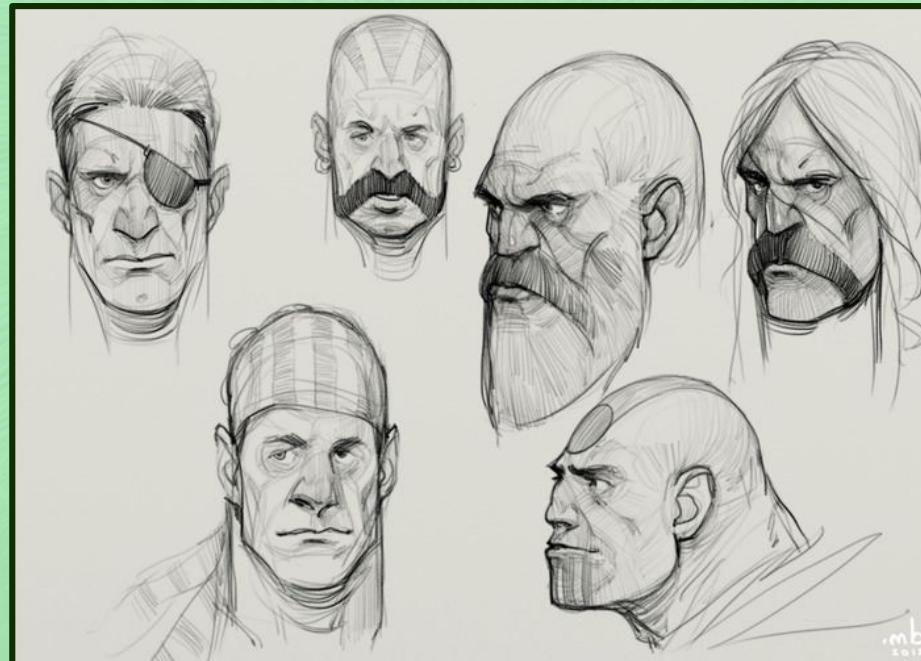
# Level Creation

- Do ponto de vista do tempo, o primeiro nível do jogo leva mais tempo para se desenvolver. À medida que designers e artistas de nível usam as ferramentas para construção de nível, eles solicitam recursos e alterações nas ferramentas internas que permitem um desenvolvimento mais rápido e de maior qualidade.



# Produção de Arte

- O design da arte do game é um subconjunto do desenvolvimento de games. É o processo de criação dos aspectos artísticos para videogames.
- Os artistas de videogame são artistas visuais envolvidos com a concepção do game e fazem esboços dos personagens, cenários, objetos, etc.



# Produção de Áudio

- O áudio do game pode ser separado em três categorias: efeitos sonoros, música e narração.
- A produção de efeitos sonoros é a produção de sons ajustando uma amostra para um efeito desejado ou replicando-o com objetos reais. Os efeitos sonoros são importantes e afetam a entrega do game.
- A música pode ser sintetizada ou tocada ao vivo.



# Produção de Áudio

- Existem várias maneiras de apresentar a música em um game:
  - ◆ A música pode ser ambiente, especialmente para períodos lentos do game, onde a música visa reforçar o clima estético e a configuração do game.
  - ◆ A música pode ser acionada por eventos no game. Por exemplo, em games como Pac-Man ou Mario, o jogador pegando power-ups aciona as respectivas partituras musicais.
  - ◆ Música de ação, como sequências de perseguição, batalha ou caça, é uma partitura rápida e que muda constantemente.
  - ◆ A música do menu, semelhante à música de créditos, cria um impacto auditivo enquanto relativamente pouca ação está ocorrendo.
- Um título de game com 20 horas de jogabilidade para um jogador pode ter cerca de 1 hora de áudio.
- Sobreposições de voz e dublagem criam interatividade na jogabilidade dos personagens. A dublagem adiciona personalidade aos personagens do game.

# Testes

- No final do projeto, a garantia de qualidade desempenha um papel significativo.
- Os testadores começam a trabalhar assim que tudo estiver jogável. Pode ser um nível ou subconjunto do software do game que pode ser usado em qualquer extensão razoável.
- No início, o teste de um game ocupa uma quantidade relativamente pequena de tempo. Os testadores podem trabalhar em vários games ao mesmo tempo.
- À medida que o desenvolvimento chega ao fim, um único game geralmente emprega muitos testadores em tempo integral. Eles se esforçam para testar novos recursos e testar a regressão dos existentes. O teste é vital para games modernos e complexos, pois mudanças simples podem levar a consequências catastróficas.

# Testes

- Neste momento, os recursos e níveis estão sendo concluídos com a maior taxa e há mais material novo a ser testado do que em qualquer outro momento do projeto. Os testadores precisam realizar testes de regressão para se certificar de que os recursos que estão em vigor há meses ainda funcionam corretamente.
- O teste de regressão é uma das tarefas vitais necessárias para o desenvolvimento de software eficaz. À medida que novos recursos são adicionados, mudanças sutis na base de código podem produzir mudanças inesperadas em diferentes partes do game.
- Essa tarefa é freqüentemente esquecida, por vários motivos. Às vezes, quando um recurso é implementado e testado, ele é considerado "funcionando" para o resto do projeto e pouca atenção é dada a testes repetidos. Além disso, os recursos adicionados posteriormente no desenvolvimento são priorizados e os recursos existentes geralmente recebem tempo de teste insuficiente. O teste de regressão adequado também é cada vez mais caro à medida que o número de recursos aumenta e geralmente não é programado corretamente.

# Testes

- Apesar dos perigos de ignorar os testes de regressão, alguns desenvolvedores e publishers de games não conseguem testar o conjunto completo de recursos do game e lançam ele com bugs. Isso pode resultar na insatisfação dos clientes e no fracasso em cumprir as metas de vendas. Quando isso acontece, a maioria dos desenvolvedores e publishers rapidamente lança patches que corrigem os bugs e tornam o game totalmente jogável novamente.



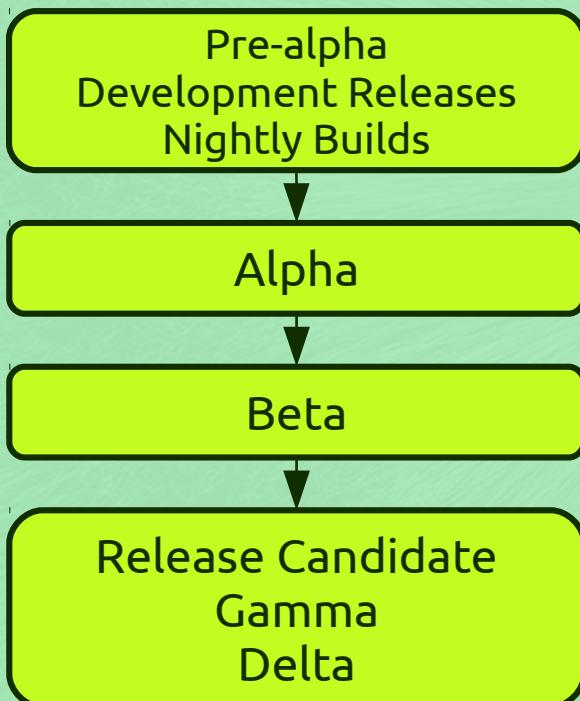
# Objetivos

- Os projetos de desenvolvimento de games comerciais podem ser necessários para cumprir os objetivos definidos pelo publishers.
- Os objetivos marcam os principais eventos durante o desenvolvimento do game e são usados para rastrear o progresso do game.
- Tais objetivos podem ser, por exemplo, primeiras versões de jogo jogáveis, alfa ou beta. Os objetivos do projeto dependem dos cronogramas do desenvolvedor.
- Os objetivos geralmente são baseados em várias descrições curtas de funcionalidade; exemplos podem ser "Jogador vagando no ambiente do jogo" ou "Física funcionando, colisões, veículo" etc. (várias descrições são possíveis).
- A lista de objetivos geralmente é um acordo colaborativo entre o publisher e o desenvolvedor. O desenvolvedor geralmente defende tornar as descrições dos objetivos o mais simples possível; dependendo do publisher específico.
- Não existe um padrão do setor para definir objetivos, e isso varia de acordo com o publisher, ano ou projeto.

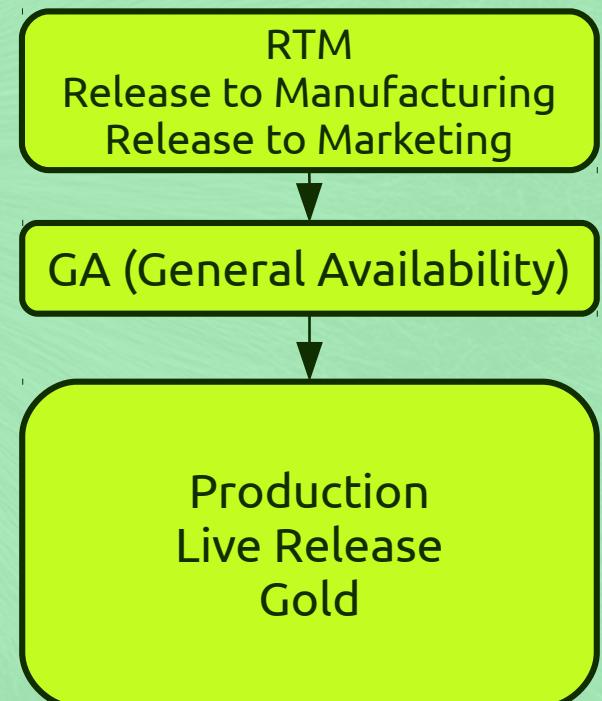
# Objetivos

- Os objetivos de desenvolvimento de videogame seguem um processo semelhante ao de outro desenvolvimento de software.

Período de Desenvolvimento e Testes



Período de Lançamento



# First Playable

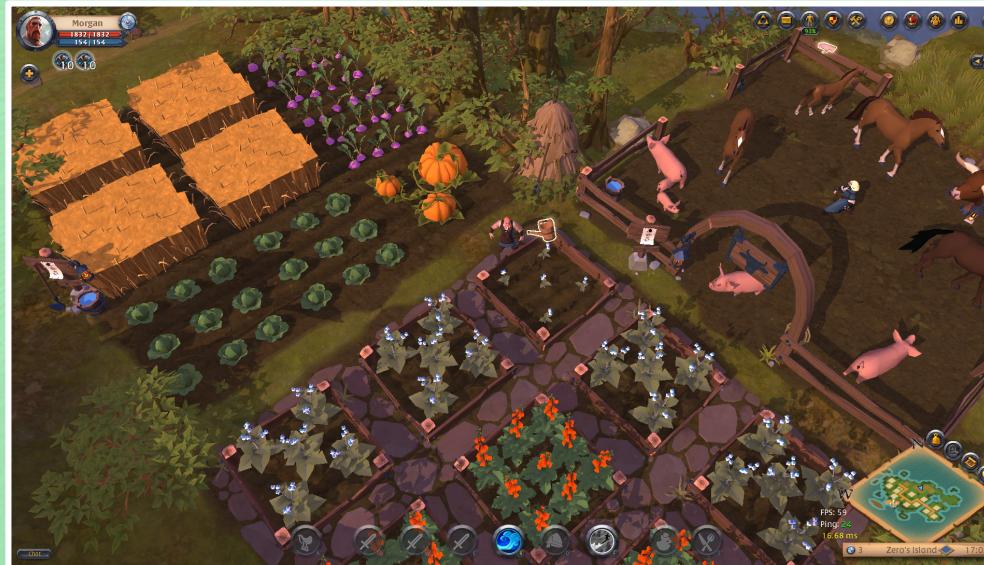
- First playable é a versão do game contendo jogabilidade e recursos representativos, esta é a primeira versão com elementos de jogabilidade principais funcionais.
- Muitas vezes é baseado no protótipo criado na pré-produção.
- Alfa e o first playable às vezes são usados para se referir a um único objetivo, no entanto, grandes projetos requerem o first playable antes de apresentar o alfa completo.
- O first playable ocorre 12 a 18 meses antes do lançamento do código.
- Às vezes é chamado de estágio "Pré-Alfa".

# Alpha

- Alpha é o estágio em que as principais funcionalidades do game são implementadas e os assets são parcialmente finalizados.
- Um game em alfa possui recursos completos, ou seja, o game pode ser reproduzido e contém todos os recursos principais.
- Esses recursos podem ser revisados posteriormente com base em testes e feedback.
- Recursos adicionais pequenos e novos podem ser adicionados, planejados de forma semelhante, mas os recursos não implementados podem ser descartados.
- Os programadores se concentram principalmente em finalizar a base de código, em vez de implementar adições.

# Beta

- Beta é uma versão completa de recursos e assets do game, quando apenas bugs estão sendo corrigidos.
- Esta versão não contém bugs que impeçam o game de ser distribuído.
- Nenhuma alteração é feita nos recursos, assets ou código do game.
- O beta ocorre dois a três meses antes do lançamento do código.



# Code Release

- A liberação do código é o estágio em que muitos bugs são corrigidos e o game está pronto para ser lançado ou enviado para análise do fabricante do console. Esta versão é testada em relação ao plano de teste de controle de qualidade.



# Gold Master

- Gold master é a construção final do game que é usada como master para a produção do game.



# Manutenção

- Assim que o game é lançado, a fase de manutenção do videogame começa.
- Games desenvolvidos para consoles de videogame quase não tiveram período de manutenção no passado. O game enviado abrigaria para sempre tantos bugs e recursos quanto quando fosse lançado. Isso era comum para consoles, pois todos os consoles tinham hardware idêntico ou quase idêntico; tornando a incompatibilidade, a causa de muitos bugs, um problema. Nesse caso, a manutenção ocorreria apenas no caso de um port, sequência ou remake aprimorado que reutilize grande parte da engine e dos assets.
- Recentemente, a popularidade dos games de console online cresceu e se desenvolveram consoles de videogame e serviços online como o Xbox Live para Xbox. Os desenvolvedores podem manter seu software por meio de patches para download. Essas mudanças não teriam sido possíveis no passado sem a ampla disponibilidade da Internet.

# Manutenção

- O desenvolvimento do PC é diferente. Os desenvolvedores de games tentam considerar a maioria das configurações e hardware. No entanto, o número de configurações possíveis de hardware e software inevitavelmente leva à descoberta de circunstâncias de quebra do game que os programadores e testadores não levaram em consideração.
- Os programadores esperam por um período para obter o maior número possível de relatórios de bug. Assim que o desenvolvedor achar que obteve feedback suficiente, os programadores começam a trabalhar em um patch. O patch pode levar semanas ou meses para ser desenvolvido, mas tem como objetivo corrigir a maioria dos bugs e problemas do game que foram esquecidos no lançamento do código anterior ou, em casos raros, corrigir problemas não intencionais causados por patches anteriores. Ocasionalmente, um patch pode incluir recursos ou conteúdo extra ou pode até alterar a jogabilidade.
- No caso de um massively multiplayer online game (MMOG), o envio do game é a fase inicial de manutenção. Esses games online estão em manutenção contínua, pois o mundo do game é continuamente alterado e iterado e novos recursos são adicionados.

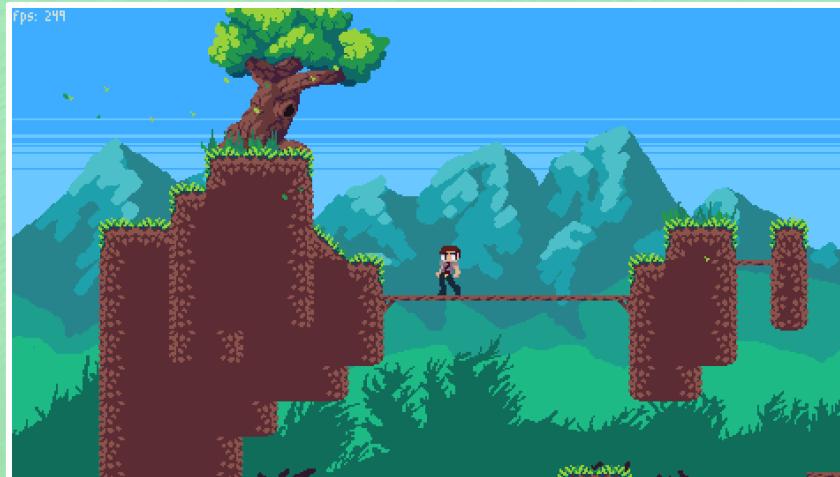
# Localização

- Um game criado em um idioma também pode ser publicado em outros países que falam um idioma diferente. Para essa região, os desenvolvedores podem querer traduzir o game para torná-lo mais acessível.
- A localização é o processo de tradução dos recursos de idioma de um game para outros idiomas. Localizando games, eles aumentam seu nível de acessibilidade.



# Indie Development

- Os independent games ou indie games são produzidos por indivíduos e pequenas equipes, sem afiliações de desenvolvedores ou publishers em grande escala.
- Os desenvolvedores independentes geralmente dependem de esquemas de distribuição na Internet. Muitos desenvolvedores indie amadores criam mods de games existentes.
- Nos últimos anos, muitas comunidades surgiram em apoio a games independentes, como o popular mercado de jogos indie **Itch.io**.



# Games de Plataforma

- Os games de plataforma são um gênero de videogame e um subgênero de games de ação em que o objetivo principal é mover o personagem do jogador entre pontos em um ambiente renderizado.



# Games de Plataforma

- Os games de plataforma são caracterizados por seu design de nível apresentando terreno irregular e plataformas suspensas de alturas variadas que requerem o uso das habilidades do personagem do jogador (como pular e escalar) para navegar pelo ambiente do jogador e alcançar seu objetivo.
- Outras manobras acrobáticas podem afetar a jogabilidade também, como escalar, balançar de objetos como videiras ou ganchos, pular de paredes, correr no ar, deslizar pelo ar, ser disparado de canhões ou saltar de trampolins.
- Embora comumente associados a games de console, há muitos games de plataforma proeminentes lançados para fliperamas, bem como para consoles de games portáteis e computadores domésticos.

# Games de Plataforma

- Um game de plataforma requer que o jogador manobre seu personagem através das plataformas para alcançar um objetivo, enquanto enfrenta os inimigos e evita obstáculos ao longo do caminho.



# Games de Plataforma

- Esses games são apresentados em vista lateral, usando movimento bidimensional, ou em 3D com a câmera posicionada atrás do personagem principal ou em perspectiva isométrica. A jogabilidade típica de plataforma tende a ser muito dinâmica e desafia os reflexos, o tempo e a destreza do jogador com os controles.
- As opções de movimento mais comuns no gênero são caminhar, correr, pular, atacar e escalar. O salto é o elemento chave da jogabilidade em todos os games de plataforma e pode ser categorizado como "comprometido" ou "variável"; em que a trajetória de um salto comprometido não pode ser alterada no ar, a de um salto variável pode. Em alguns games de plataforma, cair de alturas consideráveis pode causar danos, possivelmente até a morte.
- Muitos games de plataforma apresentam zonas dentro do mundo do game que matariam o personagem do jogador instantaneamente ao contato, como poços de lava ou abismos sem fundo. Através das várias áreas do mundo do game, o jogador pode ser capaz de coletar itens e upgrades que podem ser úteis para diferentes situações e dar ao personagem principal novas habilidades que podem tornar mais fácil para ele enfrentar as adversidades.

# Games de Plataforma

- A maioria dos games deste gênero consiste em vários níveis de dificuldade crescente, que também podem ser intercalados por encontros com chefes, onde o personagem tem que derrotar um inimigo particularmente perigoso para progredir.

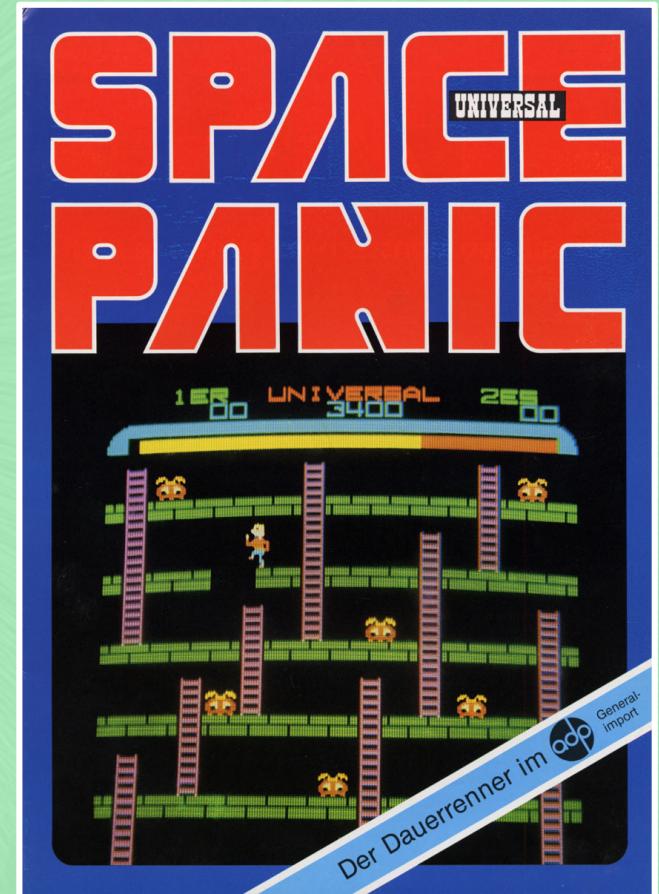


# Games de Plataforma

- Normalmente, a ordem dos níveis é pré-determinada, mas alguns games também permitem que os jogadores naveguem livremente pelo mundo do game ou podem apresentar diferentes caminhos a serem seguidos em determinados pontos.
- Quebra-cabeças lógicos simples para resolver e testes de habilidade para superar são outro elemento comum no gênero.
- O termo game de plataforma surgiu nos anos seguintes ao lançamento do primeiro título consagrado no gênero, Donkey Kong (1981), embora sua origem exata seja desconhecida. Shigeru Miyamoto originalmente chamou Donkey Kong de "game de corrida/salto/escalada" enquanto o desenvolvia.
- Miyamoto costumava usar o termo "game de atletismo" para se referir a Donkey Kong e a games posteriores do gênero, como Super Mario Bros. (1985).

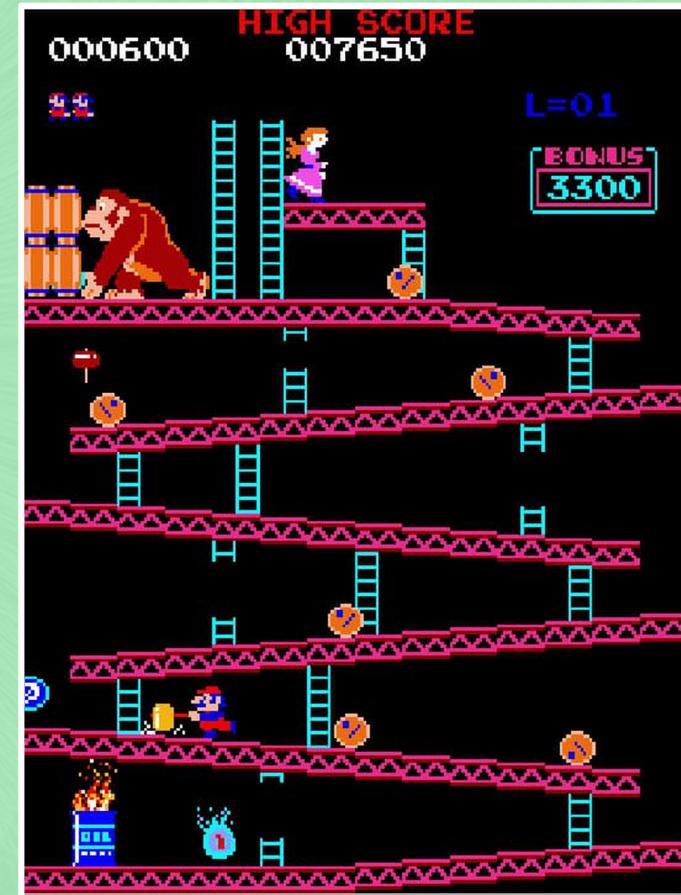
# Games de Plataforma

- Os games de plataforma surgiram no início dos anos 1980.
- A maioria dos primeiros exemplos de games de plataforma estavam confinados a um campo de jogo estático, geralmente visto de perfil, e eram baseados na mecânica de escalada entre plataformas, em vez de saltos.
- Space Panic, um lançamento de arcade de 1980 da Universal, às vezes é creditado como o primeiro jogo de plataforma.



# Games de Plataforma

- Donkey Kong, um game de arcade criado pela Nintendo e lançado em julho de 1981, foi o primeiro game a permitir aos jogadores saltar obstáculos e lacunas, explicando porque é amplamente considerado o primeiro game de plataforma, devido a essas características definidoras.
- Ele apresentou Mario sob o nome de Jumpman. Donkey Kong foi transportado para muitos consoles e computadores na época. O game ajudou a cimentar a posição da Nintendo como um nome importante no videogame indústria internacionalmente.



# Games de Plataforma

- O primeiro game de plataforma a usar gráficos de scrolling surgiu anos antes do gênero se tornar popular. Jump Bug é um game de tiro de plataforma desenvolvido por Alpha Denshi sob contrato para Hoei/Coreland e lançado para fliperamas em 1981, apenas cinco meses depois de Donkey Kong. Os jogadores controlam um carro saltitante que salta em várias plataformas, como edifícios, nuvens e colinas. Jump Bug ofereceu um vislumbre do que estava por vir, com plataformas suspensas desiguais e níveis que rolavam horizontalmente e, em uma seção, verticalmente.



# Games de Plataforma

- O Super Mario Bros. da Nintendo, lançado para o Nintendo Entertainment System em 1985, tornou-se o arquétipo de muitos games de plataforma. Foi empacotado com sistemas Nintendo na América do Norte, Japão e Europa, e vendeu mais de 40 milhões de cópias, de acordo com o Livro de Recordes Mundiais de 1999 do Guinness. Seu sucesso levou muitas empresas a ver os games de plataforma como vitais para seu sucesso e contribuiu muito para popularizar o gênero durante a geração do console de 8 bits.



# Games de Plataforma

- A Sega tentou emular esse sucesso com sua série Alex Kidd, que começou em 1986 no Master System com Alex Kidd in Miracle World. Ele tinha níveis de scrolling horizontal e vertical, a capacidade de socar inimigos e obstáculos e lojas para o jogador comprar power-ups e veículos.



# Games de Plataforma

- Em 1987, o Mega Man da Capcom introduziu a progressão de nível não linear, onde o jogador é capaz de escolher a ordem em que conclui os níveis. Isso foi um grande contraste com os games lineares como Super Mario Bros. e games de mundo aberto como Metroid.



# PyGame

- Pygame é um conjunto de módulos Python projetados para escrever videogames.
- O Pygame adiciona funcionalidade à excelente biblioteca SDL. Isso permite que você crie games completos e programas multimídia na linguagem Python.
- O Pygame é altamente portátil e funciona em quase todas as plataformas e sistemas operacionais.
- Pygame é grátis. Lançado sob a licença LGPL, você pode criar jogos de código aberto, freeware, shareware e comerciais com ele.



# PyGame

- Pygame requer Python; se ainda não o tiver em sua máquina, você pode baixá-lo em [python.org](http://python.org).
- Use o python 3.7.7 ou superior, porque é muito mais amigável para os novatos e, além disso, executa mais rápido.
- A melhor maneira de instalar o pygame é com a ferramenta **pip** (que é o que o python usa para instalar pacotes). Observe que o **pip** já vem com o Python em versões recentes.
- Usamos o sinalizador **--user** para instruí-lo a instalar no diretório **home**, em vez de globalmente.

```
python3 -m pip install -U pygame --user
```

- Para ver se funciona, execute um dos exemplos incluídos:

```
python3 -m pygame.examples.aliens
```

# PyGame

- Importar e inicializar o pygame é um processo muito simples.
- Também é flexível o suficiente para lhe dar controle sobre o que está acontecendo.
- Pygame é uma coleção de diferentes módulos em um único pacote Python.
- Alguns dos módulos são escritos em C, e alguns são escritos em Python.
- Alguns módulos também são opcionais e nem sempre estão presentes.
- Primeiro, devemos importar o pacote pygame. A maioria dos games importará todo o pygame assim.

```
import pygame  
from pygame.locals import *
```

- A primeira linha aqui é a única necessária. Ele importa todos os módulos pygame disponíveis para o pacote pygame. A segunda linha é opcional e coloca um conjunto limitado de constantes e funções no namespace global do seu script.

# PyGame

- A instrução de importação do pygame é sempre colocada no início do programa.
- Ela importa as classes, métodos e atributos do pygame para o espaço de nomes atual.
- Agora, esses novos métodos podem ser chamados via **pygame.method()**.
- Por exemplo, agora podemos inicializar ou sair do pygame com os seguintes comandos:

```
pygame.init()  
pygame.quit()
```

- **Init** tentará inicializar todos os módulos do pygame para você. Nem todos os módulos do pygame precisam ser inicializados, mas isso inicializará automaticamente aqueles que precisam.
- Você também pode inicializar facilmente cada módulo do pygame manualmente. Por exemplo, para inicializar apenas o módulo de fonte você apenas chamaria.

```
pygame.font.init()
```

# PyGame

- **Surface:** A parte mais importante do pygame é a surface. Pense em uma surface como um pedaço de papel em branco.
- Você pode fazer muitas coisas com uma surface - você pode desenhar linhas nela, preencher partes dela com cores, copiar imagens de e para ela e definir ou ler cores de pixel individuais nela.
- Uma surface pode ter qualquer tamanho (dentro do razoável) e você pode ter quantos delas quiser (novamente, dentro do razoável).
- Há uma surface que é especial - aquela que você cria com **pygame.display.set\_mode()**.
- Esta 'superfície de exibição' representa a tela; tudo o que você fizer aparecerá na tela do usuário. Você pode ter apenas uma dessa, e esta é uma limitação do SDL, não do pygame.

# PyGame

- Uma surface de desenho tem um tamanho específico, que é definido quando a criamos.
- O tamanho é dado em pixels (um pixel é o tamanho de um ponto na tela). Quanto mais pixels você tiver, melhor será a qualidade da tela.
- Usaremos um tamanho de tela de 800 pixels de largura e 600 pixels de altura.
- Podemos usar uma tupla para criar uma superfície da seguinte maneira:

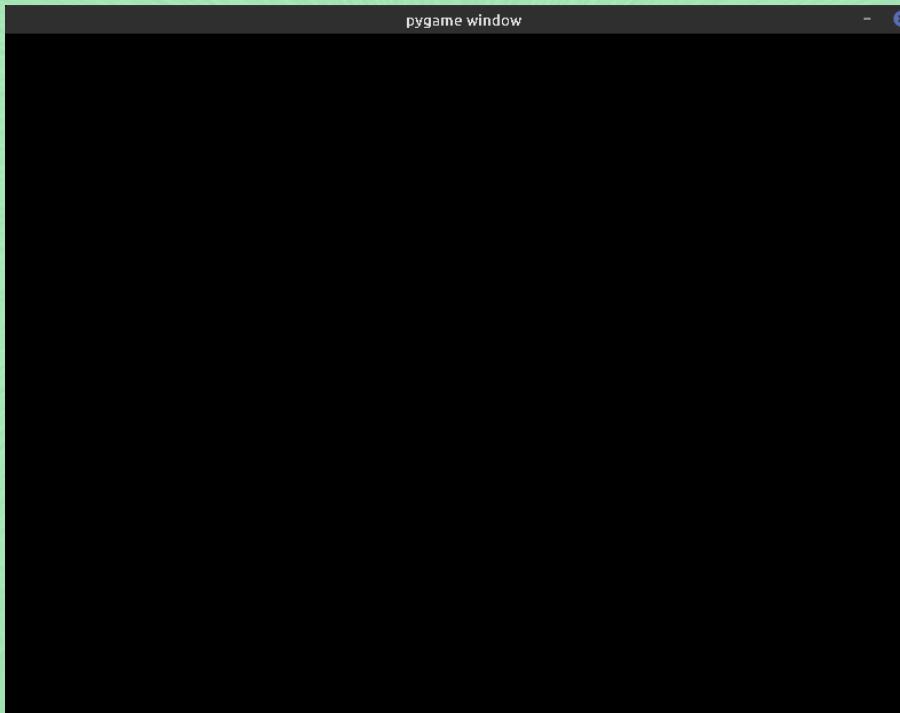
```
tamanho = (800, 600)
```

- Assim que tivermos a tupla que descreve o tamanho da tela do jogo, podemos usar esse valor como um argumento para a função que cria uma superfície de desenho de pygame.

```
surface = pygame.display.set_mode(tamanho)
```

# PyGame

- O comando anterior cria a **surface** de desenho, define a variável **surface** para se referir a ela e, em seguida, exibe a **surface** na tela.
- Você deve ver a seguinte janela aparecer na sua tela.



# PyGame

- Você pode alterar o título da janela de desenho usando a seguinte função:

```
pygame.display.set_caption('Meu Primeiro Game')
```
- Então, como você cria **surfaces**?
- Como mencionado anteriormente, você cria a 'surface de exibição' especial com **pygame.display.set\_mode()**.
- Você pode criar uma surface que contém uma imagem usando o método **image.load()** ou pode fazer uma surface que contém texto com **font.render()**. Você pode até mesmo criar uma surface que não contém nada com **Surface()**.
- A maioria das funções de surface não são críticas. Apenas aprenda **blit()**, **fill()**, **set\_at()** e **get\_at()**, e você ficará bem.

# PyGame

- Para desenhar uma imagem na **surface principal**, carregamos a imagem, digamos um feiticeiro, em sua própria **nova surface**.
- A surface principal tem um método **blit** que copia pixels da surface do feiticeiro em sua própria surface. Quando chamamos **blit**, podemos especificar onde o feiticeiro deve ser colocado na surface principal.
- O termo **blit** é amplamente usado em computação gráfica e significa fazer uma cópia rápida de pixels de uma área da memória para outra.
- Portanto, na seção de configuração, antes de entrar no **Game Loop**, carregamos a imagem, assim:

```
feiticeiro = pygame.image.load("wizard.png")
```

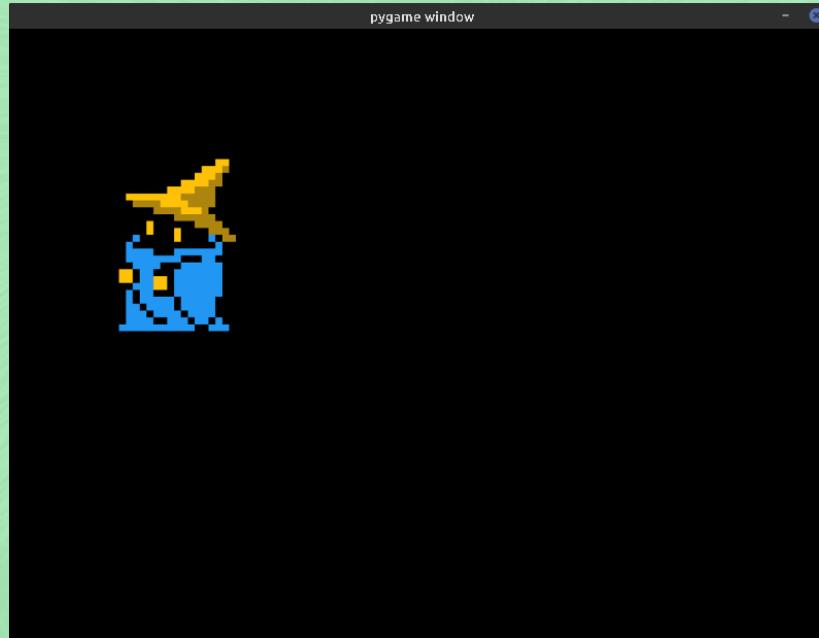
- E para apresentar ele na posição (100, 120), usamos o método **blit**:

```
surface.blit(feiticeiro, (100, 120))
```

# PyGame

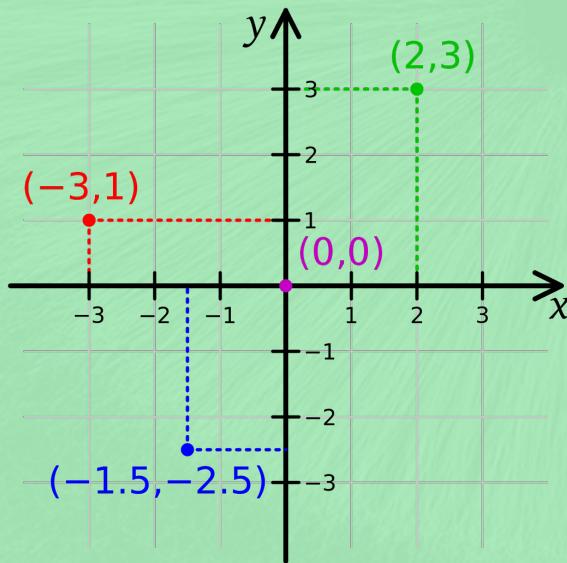
- Observe que nosso feiticeiro não irá aparecer imediatamente, para que isso ocorra, precisamos atualizar a tela com o método **update**.

```
pygame.display.update()
```



# PyGame

- Talvez você tenha estranhado o local ao qual a nossa imagem apareceu, isso ocorre pelo fato de que o sistema de coordenadas do pygame é um pouco diferente do sistema de coordenadas cartesiano tradicional, que é o sistema ao qual a maioria das pessoas está acostumada ao plotar gráficos. Este é o sistema ensinado nas escolas. O pygame usa um sistema de coordenadas semelhante, mas um pouco diferente.

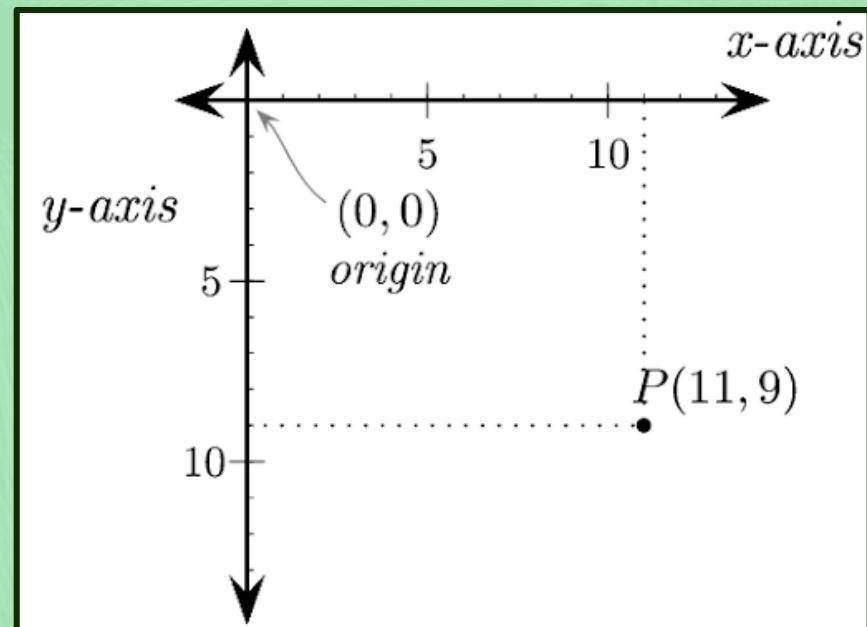


# PyGame

- No pygame as coordenadas x funcionam da mesma forma que o sistema de coordenadas cartesianas, porém as coordenadas y estão invertidas. Em vez da coordenada zero y na parte inferior do gráfico, como nos gráficos cartesianos, a coordenada zero y está na parte superior da tela com o computador. À medida que os valores de y aumentam, a posição das coordenadas do computador desce na tela.

Além disso, observe que a tela cobre o quadrante inferior direito, onde o sistema de coordenadas cartesianas geralmente se concentra no quadrante superior direito.

É possível desenhar itens em coordenadas negativas, mas eles serão desenhados fora da tela. Isso pode ser útil quando parte de uma forma está fora da tela. O computador descobre o que está fora da tela e o programador não precisa se preocupar muito com isso.



# PyGame

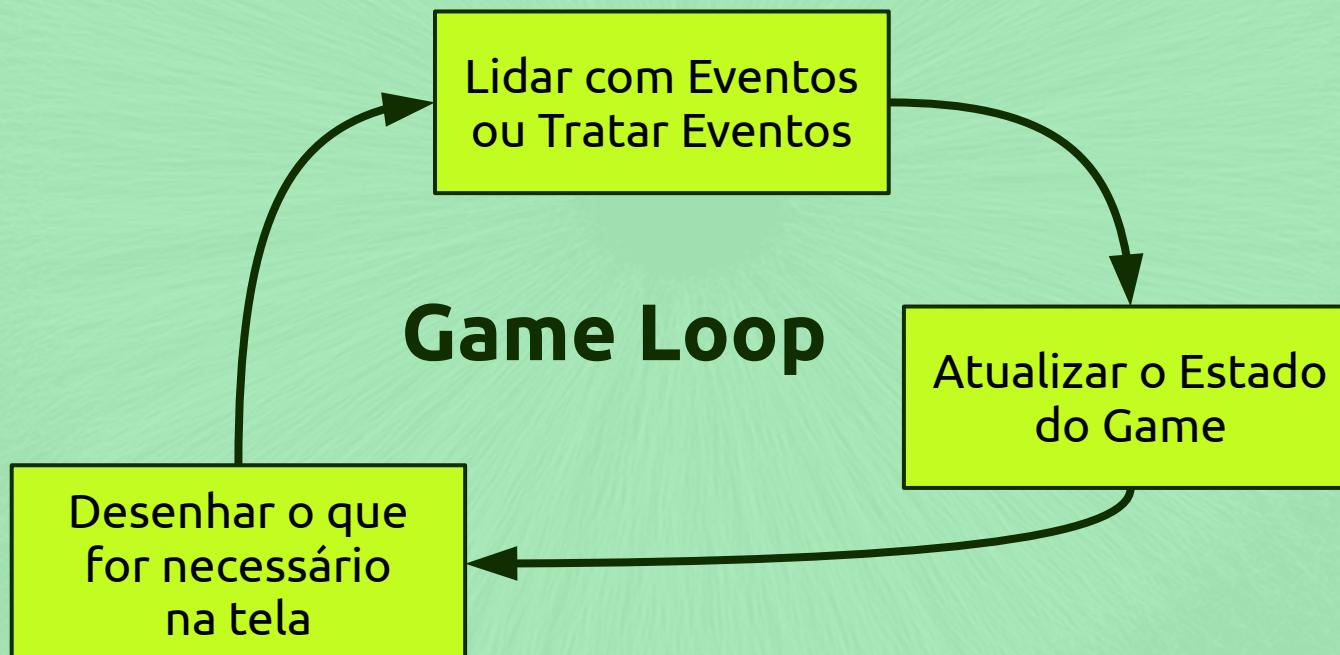
- **Game Loop:** Um game loop (também chamado de loop principal) é um loop em que o código faz três ações principais:
  1. Lida com eventos.
  2. Atualiza o estado do game.
  3. Desenha o estado atual do game na tela.
- O **estado do game** é simplesmente uma forma de se referir a um conjunto de valores para todas as variáveis em um programa de game. Em muitos games, o estado do game inclui os valores nas variáveis que rastreiam a vida e a posição do jogador, a vida e a posição de quaisquer inimigos, quais marcas foram feitas em um tabuleiro, a pontuação ou de quem é a vez.
- Sempre que algo acontece, como o jogador sofrendo dano (o que diminui seu valor de vida), ou um inimigo se move para algum lugar, ou algo acontece no mundo do game, dizemos que o estado do game mudou.

# PyGame

- Se você já jogou um game que permitiu salvar ele, o "estado de salvamento" é o estado do game no ponto em que você o salvou. Na maioria dos games, pausar o game impedirá que o estado dele mude.
- Como o estado do game é normalmente atualizado em resposta a eventos (como cliques do mouse ou pressionamentos de teclado) ou à passagem do tempo, o game loop está constantemente verificando e re-verificando novamente muitas vezes por segundo para quaisquer novos eventos que tenham acontecido.
- Dentro do loop principal está o código que verifica quais eventos foram criados (com pygame, isso é feito chamando a função **pygame.event.get()**).
- O loop principal também possui um código que atualiza o estado do game com base nos eventos criados. Isso geralmente é chamado de tratamento de eventos.

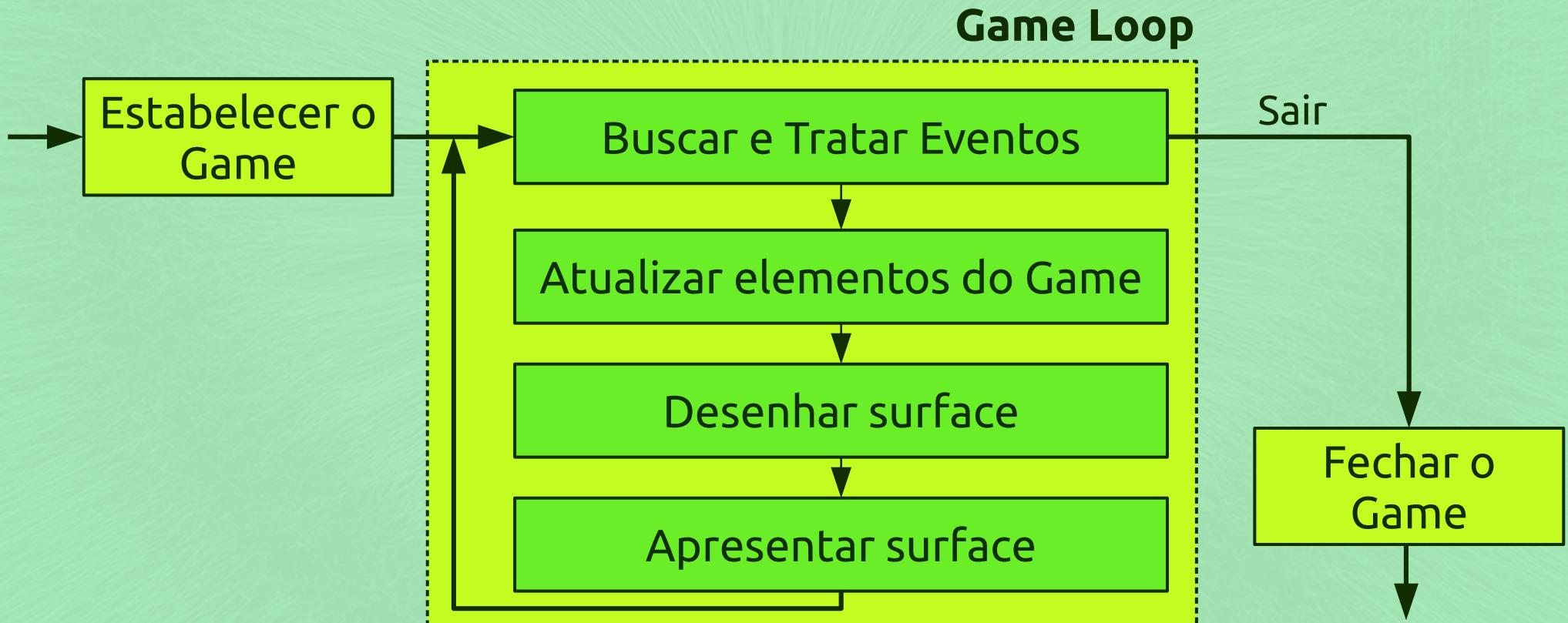
# PyGame

- O esquema a seguir nos fornece uma ideia do comportamento do **Game Loop**:



# PyGame

- A estrutura dos games que consideraremos irá seguir este padrão fixo:



# PyGame

- A parte mais essencial de qualquer aplicativo interativo é o loop de eventos.
- Reagir a eventos permite que o usuário interaja com o aplicativo.
- Eventos são coisas que podem acontecer em um programa, como:
  - ◆ Clique de mouse.
  - ◆ Movimento do mouse.
  - ◆ Tecla do teclado pressionada.
  - ◆ Ação do joystick.
- A seguir temos um loop infinito que imprime todos os eventos no console:

```
while True:  
    for event in pygame.event.get():  
        print(event)
```

# PyGame

- Tente mover o mouse, clicar em um botão do mouse ou digitar algo no teclado.
- Cada ação que você faz produz um evento que será impresso no console.
- Como estamos em um loop infinito, é impossível sair deste programa de dentro do aplicativo.
- Para sair do programa, torne o console a janela ativa e digite **CTRL + C**.
- Para encerrar o aplicativo corretamente, de dentro do aplicativo, usando o botão Fechar da janela (evento **QUIT**), modificamos o loop de eventos.
- A seguir veremos como fazer essa modificação.

# PyGame

- Primeiro, apresentamos a variável booleana **running** e a definimos como **True**.
- No loop de eventos, verificamos o evento **QUIT**. Se ele ocorrer, definimos **running** como **False**:

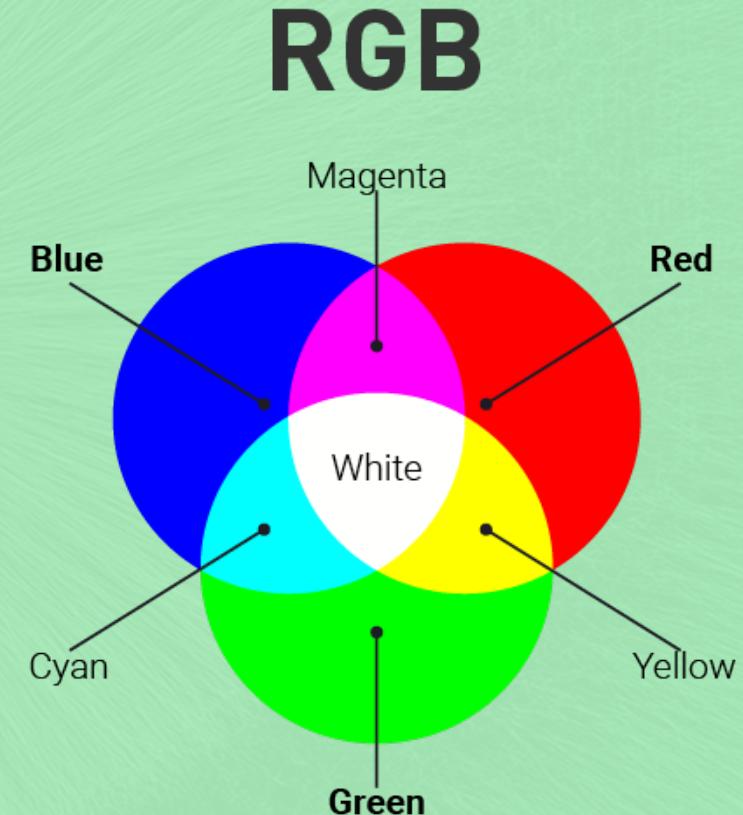
```
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

pygame.quit()
```

- Após o loop de eventos, chamamos a função **pygame.quit()** para encerrar o aplicativo corretamente, isso quer dizer que se o usuário clicar para fechar a janela, **running** se tornará **False**, o loop irá encerrar, e a janela irá fechar.

# PyGame

- As cores são definidas como tuplas das cores básicas vermelho, verde e azul. Isso é chamado de **modelo RGB**.
- Cada cor de base é representada como um número entre 0 (mínimo) e 255 (máximo) que ocupa 1 byte na memória.
- Uma cor RGB é então representada como um valor de 3 bytes.
- A mistura de duas ou mais cores resulta em novas cores. Um total de 16 milhões de cores diferentes podem ser representadas dessa forma.



# PyGame

- Vamos definir as cores básicas como tuplas dos três valores básicos.
- Como as cores são constantes, vamos escrevê-las em maiúsculas.
- A ausência de todas as cores resulta em preto.
- O valor máximo para todos os três componentes resulta em branco.
- Três valores intermediários idênticos resultam em cinza:

PRETO = (0, 0, 0)

CINZA = (127, 127, 127)

BRANCO = (255, 255, 255)

- As três cores base são definidas como:

VERMELHO = (255, 0, 0)

VERDE = (0, 255, 0)

AZUL = (0, 0, 255)

# PyGame

- Ao misturar duas cores de base, obtemos mais cores:

```
AMARELO = (255, 255, 0)  
CIANO = (0, 255, 255)  
MAGENTA = (255, 0, 255)
```

- Se quisermos preencher a tela com uma cor, podemos usar o método **fill**.

```
surface.fill(BRANCO)  
pygame.display.update()
```

- O método **fill(cor)** preenche toda a tela com a cor especificada.
- Pygame também conta com cores imbutidas, que podemos acessar da seguinte maneira.

```
print(pygame.color.THECOLORS)
```

# PyGame

- O retângulo é um objeto muito útil na programação gráfica.
- Ele tem sua própria classe **Rect** no Pygame e é usado para armazenar e manipular uma área retangular.
- Um objeto Rect pode ser criado fornecendo:
  - ◆ Os 4 parâmetros **left**, **top**, **width** e **height**.
  - ◆ A **position** e **size**.
  - ◆ Um **objeto** que tem um atributo **rect**.

```
Rect(left, top, width, height)
```

```
Rect(pos, size)
```

```
Rect(obj)
```

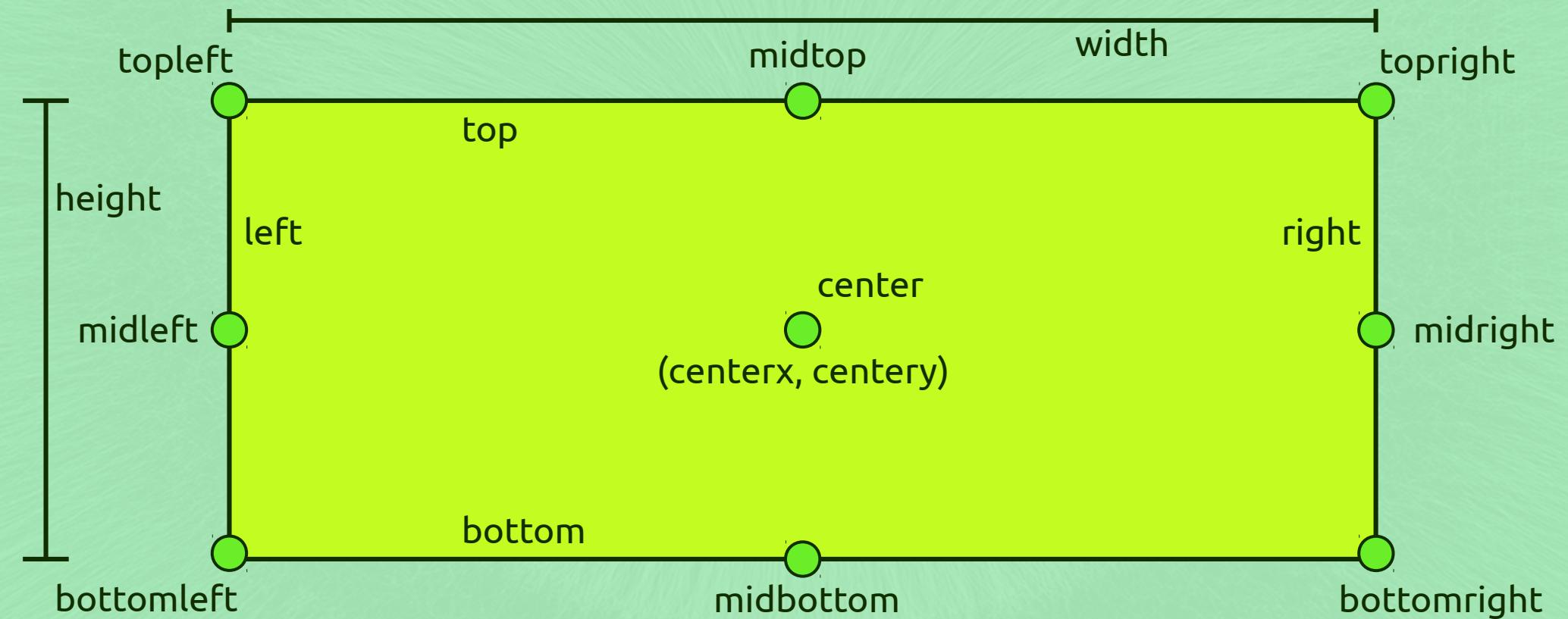
# PyGame

- Uma função que espera um argumento Rect aceita igualmente um dos três valores anteriormente apresentados.
- Métodos que mudam a posição ou tamanho, como **move()** e **inflate()** deixam o Rect original intocado e retornam um novo Rect.
- Eles também têm as versões **move\_ip** e **inflate\_ip** que atuam sobre o Rect original.
- O objeto Rect tem vários atributos virtuais que podem ser usados para mover e alinhar o Rect. A atribuição a esses atributos apenas move o retângulo sem alterar seu tamanho:

```
x, y  
top, left, bottom, right  
topleft, bottomleft, topright, bottomright  
midtop, midleft, midbottom, midright  
center, centerx, centery
```

# PyGame

- A seguir podemos ver graficamente os diferentes atributos do Rect:



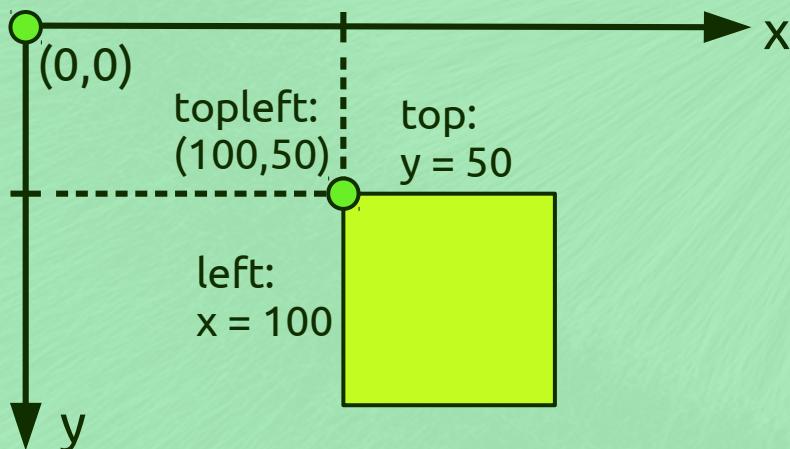
# PyGame

- A atribuição desses 5 atributos altera o tamanho do retângulo, mantendo sua posição superior esquerda.

size, width, height, w, h

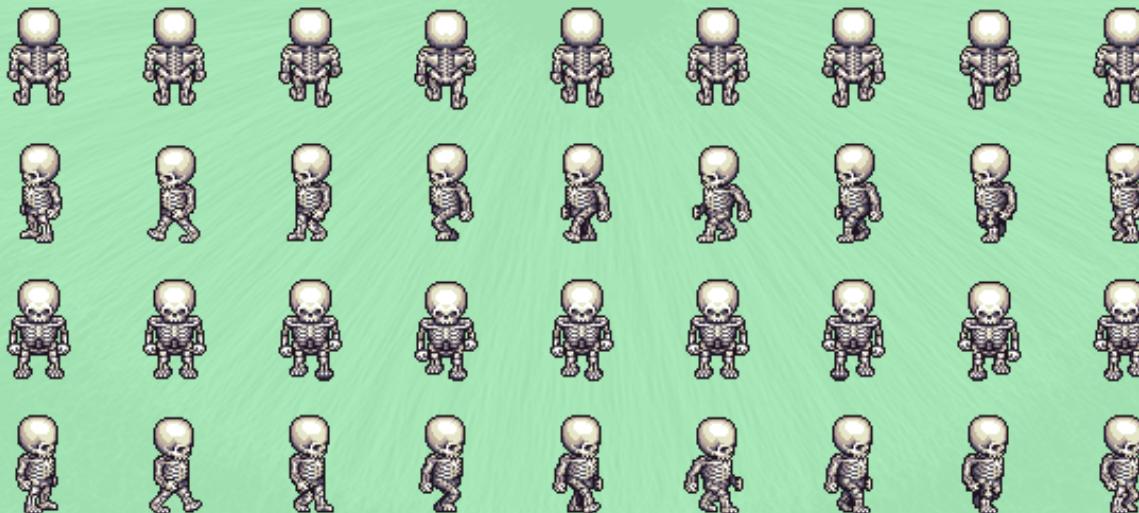
- O retângulo a seguir foi criado na posição (100,50) da tela e possui 20x30 pixels.

```
retângulo = pygame.Rect(100,50,20,30)
```



# PyGame

- Em computação gráfica, um **sprite** é um bitmap bidimensional integrado em uma cena maior, na maioria das vezes em um videogame 2D.
- Originalmente, o termo sprite se referia a objetos compostos juntos de tamanho fixo, por hardware, com um fundo.
- Desde então, o uso do termo tornou-se mais geral.



# PyGame

- O Pygame versão 1.3 vem com um novo módulo, **pygame.sprite**.
- Este módulo foi escrito em Python e inclui algumas classes de nível superior para gerenciar seus objetos de game. Usando este módulo em todo o seu potencial, você pode gerenciar e desenhar facilmente seus objetos. As classes sprite são muito otimizadas, então é provável que seu game rode mais rápido com o módulo sprite do que sem.
- O módulo sprite vem com duas classes principais.
- O primeiro é **Sprite**, que deve ser usado como uma classe base para todos os seus objetos de game. Esta classe realmente não faz nada por conta própria, apenas inclui várias funções para ajudar a gerenciar o objeto do game.
- O outro tipo de classe é o **Group**. A classe Group é um contêiner para diferentes objetos Sprite. Na verdade, existem vários tipos diferentes de classes em group. Alguns dos Groups podem desenhar todos os elementos que contêm, por exemplo.

# PyGame

- A classe Sprite foi projetada para ser uma classe base para todos os seus objetos de game.
- Você não pode realmente usá-la sozinha, pois ela só tem vários métodos para ajudá-lo a trabalhar com as diferentes classes de Group.
- O sprite mantém registro de quais grupos ele pertence.
- O construtor de classe (método `__init__`) recebe um argumento de um Group (ou lista de Groups) ao qual a instância Sprite deveria pertencer.
- Você também pode alterar a associação de Group para o Sprite com os métodos `add()` e `remove()`.
- Também existe um método `groups()`, que retorna uma lista dos grupos atuais contendo o sprite.

# PyGame

- Ao usar as classes de Sprite, é melhor pensar nelas como "válidas" ou "vivas" quando pertencem a um ou mais Groups.
- Quando você remove a instância de todos os groups, o pygame limpa o objeto. (A menos que você tenha suas próprias referências à instância em outro lugar.)
- O método **kill()** remove o sprite de todos os grupos aos quais ele pertence. Isso excluirá o objeto sprite de forma limpa.
- Se você montou alguns pequenos games, sabe que às vezes pode ser complicado excluir um objeto do game de forma limpa.
- O sprite também vem com um método **alive()**, que retorna True se o sprite ainda for membro de algum grupo.

# PyGame

- A classe Group é apenas um contêiner simples.
- Semelhante ao sprite, ele tem um método **add()** e **remove()** que podem alterar quais sprites pertencem ao grupo.
- O Group tem alguns outros métodos como **empty()** para remover todos os sprites do grupo e **copy()** que retornará uma cópia do group com todos os mesmos membros.
- Além disso, o método **has()** irá verificar rapidamente se o Group contém um sprite ou uma lista de sprites.
- A outra função que você usará com frequência é o método **sprites()**. Isso retorna um objeto que pode ser executado em loop para acessar todos os sprites que o group contém.
- Como um atalho, o Group também possui um método **update()**, que irá chamar um método **update()** em cada sprite no grupo. Passando os mesmos argumentos para cada um.

# PyGame

- A seguir iremos usar o poder dos sprites em pygame para definir uma classe que representa um Alien.

```
class Alien(pygame.sprite.Sprite):
    def __init__(self, x, y):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load('imagens/alien.png').convert_alpha()
        self.rect = self.image.get_rect()
        self.rect.center = (x, y)
```

- Também vamos definir um método update que irá teleportar o Alien para um local aleatório da tela (neste caso, devemos importar a biblioteca **random**).

```
def update(self):
    novo_x = random.randrange(20,LARGURA - 100)
    novo_y = random.randrange(20,ALTURA - 150)
    self.rect.x = novo_x
    self.rect.y = novo_y
```

# PyGame

- E então criamos um grupo que poderá conter aliens, criamos uma instância de alien e adicionamos ela ao nosso grupo.

```
alien_group = pygame.sprite.Group()  
alien = Alien(250, 250)  
alien_group.add(alien)
```

- Para desenhar o alien em nossa tela, chamamos o método **draw()**.
- E para fazer o alien se teleportar, devemos chamar o método **update()**.

```
alien_group.draw(surface)
```

```
alien_group.update()
```



# PyGame

- O módulo sprite também vem com duas funções de detecção de colisão muito genéricas.
- Para games mais complexos, eles realmente não funcionarão para você, mas você pode facilmente pegar o código-fonte para eles e modificá-los conforme necessário.
  - ◆ **spritecollide(sprite, group, dokill)**: verifica se há colisões entre um único sprite e os sprites de um group. Requer um atributo "rect" para todos os sprites usados. Ele retorna uma lista de todos os sprites que se sobrepõem ao primeiro sprite. O argumento "dokill" é um argumento booleano. Se for True, a função irá chamar o método **kill()** em todos os sprites.
  - ◆ **groupcollide(group1, group2, dokill1, dokill2)**: ele verifica se há colisões para todos os sprites de um grupo e para os sprites de outro. Há um argumento dokill para os sprites em cada lista. Quando "dokill1" for verdadeiro, os sprites em colisão no group1 serão **kill()**. Quando "dokill2" é verdadeiro, obtemos os mesmos resultados para o group2. O dicionário que ele retorna funciona assim; cada chave no dicionário é um sprite do group1 que teve uma colisão. O valor dessa chave é uma lista dos sprites com os quais ela colidiu.

# PyGame

- A seguir temos um template básico de uma aplicação pygame.

```
import pygame
pygame.init()

WIDTH, HEIGHT = (700, 500)
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Meu Game")
clock = pygame.time.Clock()

rodando = True
while rodando:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            rodando = False
    pygame.quit()
```

# PyGame

- Para conhecer mais exemplos, visite os seguintes links:

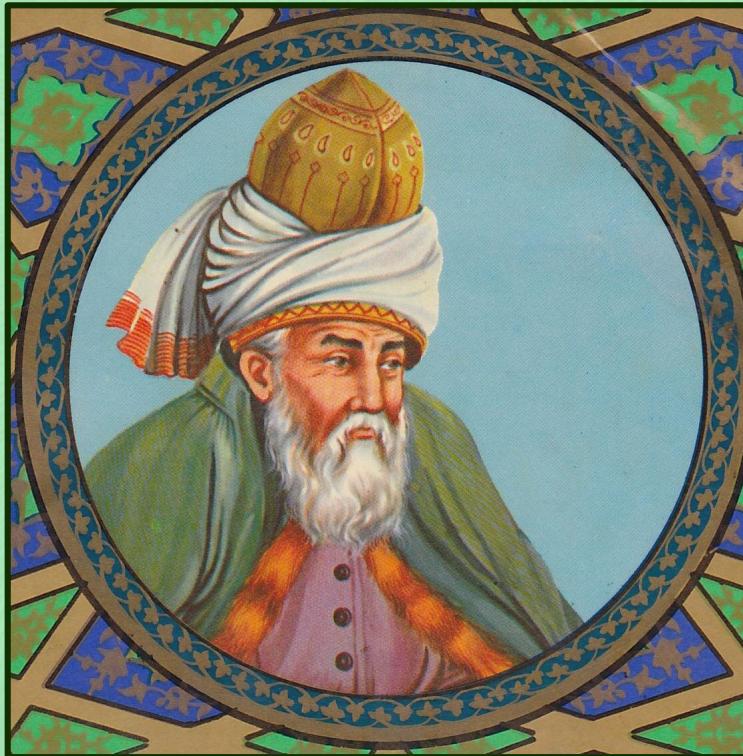


<https://akiradev.netlify.app/posts/desenvolvimento-games-python/>

<https://github.com/the-akira/PyGameDev/tree/master/Exemplos>

# Bons Estudos!

“Don't be satisfied with stories, how things have gone with others. Unfold your own myth.” **Rumi**



# Referências

- <http://inventwithpython.com/pygame/>
- <http://programarcadegames.com/>
- <https://pygame.readthedocs.io/en/latest/index.html>
- <https://www.pygame.org/docs/>
- [https://en.wikipedia.org/wiki/Video\\_game\\_development](https://en.wikipedia.org/wiki/Video_game_development)
- [https://en.wikipedia.org/wiki/Platform\\_game](https://en.wikipedia.org/wiki/Platform_game)
- <https://openbookproject.net/thinkcs/python/english3e/>