Dio programa games

@rafaskoberg

Criou jogos para minecraft e roblox

Está criando o jogo Grashers

Criou a empresa de desenvolvimento Patosaur

-

AbraGAMES – indústria brasileira de games

Unreal e unit são as principais engines.

Área de Entretenimento está em alta.

# Carreiras na área de games

1. Programação

-unity

-java

-javascript

-lua

1. Arte gráfica
2. Game design – maestro do jogo, sabe o que o jogador deve sentir.
3. Áudio design – áudio do jogo
4. Narrativa – qual a história do jogo
5. Tester (QA) – testa a mecânico do jogo.
6. Produção – trabalha na execução do projeto, um manager quase. Monitora tudo do projeto.
7. Marketing. – arte de apresentar o projeto para o público.

Programação de jogos

1. Gameplay – programa a mecânica do jogo.
2. Ai -
3. Network – programação de redes.
4. Ferramentas – criação de ferramentas próprias do projeto.
5. Gfx/shaders

Arte gráfica

1. Tech art – uma área bem especifica
2. Arte 2d e 3d
3. Animação
4. Arte conceitual – ajuda no início do projeto
5. Ui e ux - interface do usuário.

Audio Design

1. Efeitos sonos
2. Trilhas sonoras
3. Programação de áudio – conexão do áudio com a programação

Game Design

1. Prototipagem – faz algo bem simples pra verificar uma ideia.
2. Arquitetura dos sistemas –faz a lógica do jogo
3. Idealização do jogo
4. Balanceamento – equilibra a dificuldade do jogo.
5. Ux – cuida da experiência do usuário.
6. Level design – arte de criar o nível a estrutura do level que o jogar vai jogar.

**Aonde achar vagas de emprego na área de games**

[**https://academiapme-my.sharepoint.com/:p:/g/personal/nubia\_dio\_me/EYIvfAJEEAJKsDCb5JJM3HEB5a5vqt1jvYaIOtaiifuxLQ?e=EPNNWn**](https://academiapme-my.sharepoint.com/:p:/g/personal/nubia_dio_me/EYIvfAJEEAJKsDCb5JJM3HEB5a5vqt1jvYaIOtaiifuxLQ?e=EPNNWn)

Abra games – pesquisar quais empresas existem. Em afiliados e parceiros.

Ex: kokku, madmimic, minimal games, rogue snail, wild life.

The Gamer inside Brasil – idealizado jesus Fabre(divulgador da indústria de games no brasil)– pesquisar vagas aqui.

Analisar as vagas e identificar o que você consegue fazer bem. Oque não saber você pode pesquisar e tentar ganhar um conhecimento mínimo no assunto.

# Por onde começar na área de games

Estudar Estudar Estudar Estudar Estudar Estudar Estudar Estudar Estudar Estudar Estudar.

Estudar e fazer jogos, equilíbrio.

Portfolio é necessário.

1. **Github** pra programação. ex: camila Cavalcante.
2. **Artstation** para artistas.
3. **Itch.io** pra jogos finalizados.
4. **Sites próprios** para portfólios profissionais.
5. **Notion** para portfólios flexíveis. – Amora B – criadora do celeste.

**Dica pra minha carreira**

1. Sempre ter um objetivo claro para a sua carreira.
2. Montar plano de aprendizado:
   1. Definir o objetivo final
   2. Colocar um passo a passo que deve seguir pra te colocar mais perto do objetivo.
   3. Inspirações e motivações. Qual o seu motivo pra estudar?
3. Publicar jogos:
   1. Desenvolver protótipos.
   2. Finalizar projetos do começo ao fim
   3. Sempre manter o escopo do projeto pequeno. Fazer algo pequeno, não precisa fazer o jogo completo.
4. Tenha um portfolio
5. Networking – conversar com pessoas que já estão na área.

Visão geral do processo

Rami Ismail – escreveu o seguinte processo:

1. Idealização
2. Protótipo – faz do jeito mais feio e de forma rápida. Tempo ideal 2 semanas.
3. Fatia vertical (vertical slice) – desenvolver um fatia do jogo, 1% da mecânica e mais belo que o protótipo. Tem como objetivo validar uma ideia já testada e definir a direção do projeto.
4. Início da produção – agora zeramos e começamos novamente a desenvolver o jogo.
5. Alpha – significa que as funcionalidades do jogo estão prontas. Não precisa está completo
6. Beta – o jogo já tem funcionalidade e tem o restante do jogo montado.
7. Versão final – corrigir erros, bugs, deixar mais bonito.
8. Lançamento – trabalhar as integrações com a plataforma que será lançada.

# Introdução ao desenvolvimento e criação de jogos

Elementos básicos de um jogo

https://academiapme-my.sharepoint.com/:p:/g/personal/nubia\_dio\_me/EcLOJ3auRaFMu2VWh0RDonUBb6xAx3hdkKYFnPePRFvL2w?e=edl0ww

Analisar um jogo, ideia principal, os elementos e mecânicas do jogo.

Oque é game design?

Responsável por criar, arquitetar e desenvolver o jogo.

1. Idealização e protótipo do projeto.
2. Fatia vertical – algo ainda mais rustico.
3. Início da produção
4. Alpha
5. Beta – lidamos com o jogo horizontalmente completo
6. Versão final
7. Lançamento

Identidade visual em jogos

Principais plataformas e tecnologias

Unity(usado por 83% das empresas), unreal (23% usam )e godot (7%)– essas são as maiores. Você precisa saber bem uma plataforma. Caso você não conheça outra linguagem para a vaga, você pode conseguir a vaga mesmo assim.

Melhor plataforma é a que você consegue usar.

Procurar uma plataforma que use javascript na sua criação.

Os estúdios brasileiros tem preferência pela “Construct”.

Aonde Publicar

Ex: playstore(google), appstore(apple), steam(valve), web(congregate, gamejollt), consoles,itch.io.

Steam domina 98% dos jogos. Paga 100$ por jogo e dá uma porcentagem por jogo.

Direito autoral

Pestrar atenção nos direitos autorais dos itens que você baixa quando for colocar no seu jogo.

Atenção com a AI que você irá utilizar. Não utilize para evitar problemas autorais.

Respeito acima de tudo. Respeito com as pessoas que irão jogar o seu jogo.

# Versionamento de código com git e github

<https://github.com/elidianaandrade/dio-curso-git-github>

<https://academiapme-my.sharepoint.com/:p:/g/personal/renato_dio_me/EYjkgVZuUv5HsVgJUEPv1_oB_QWs8MFBY_PBQ2UAtLqucg?e=262HGK>

<https://community.chocolatey.org/packages/git>

<https://gitfluence.com> inteligência AI que ajuda tirando duvidas sobre o git.

<https://myoctacat.com>

<https://docs.github.com/en/pages/getting-started-with-github-pages>

Prof. Elidianaandrade

**O que Versionamento de código?**

Eliminar erros caso de trabalhar em equipe.

Pra arrumar esse problema, foi criado os sistemas de controle de versionamento:

Registra histórico do projeto

Gerencia quais foram as alterações

Organiza , controle e segurança.

**Sistemas de controle:**

1. CVCS centralizado cvs e subversion um servidor único responsável por todas as versões e
2. DVCS Distribuidos git e mercurial. Nesse meio os bancos de dados são distribuídos pelos usuários que estão utilizando. Você clona para sua máquina formando um “backup”, possibilita um fluxo de trabalho flexível, possibilita trabalhar sem conexão a rede.

**Git**

É um sistema de controle distribuído (gratuito). Leve e rápido.

Relacionado no início com o núcleo do Linux – bitkeeper, mas depois ele consegue a licença e muda de nome.

Comandos e sequencias da estrutura GIT:

1. Fazer um clone (git clone url) do repositório remoto
2. Usar o git commit –m pra registrar as suas alterações
3. Git pull – verifica quais foram as alterações no projeto. Precisa realizar antes de fazer o push.
4. Git push – envia os commits para o repositório remoto.

Instalando o Git:

1. Ir no site do git e fazer o download.
2. Instalar – padrão

**Configurações**

1. Abrir o gitbash

Ordem de entendimento da primeira linha no topo: nome, nome do computador, nome do compilador e por último o endereço do diretório que estamos dentro.

Para alterar o tema é só clicar com o botão da direita e ir em setings.

1. Git config --global user.name ”Rodrigo Santos”
2. Git config –global user.email “rodrigo@gmail.com”
3. Git config user.name e git config user.email – mostra o nome e o email que está configurado.
4. Git config init.defaultBranch – mostra qual é a Branch padão.
5. Git config --global --list – mostra todas as configurações globais.

**Github**

É uma plataforma pra hospedagem de código. Mascote Octocat. Servidor.

Criado em 2008, foi atacado em 2018 e nesse período a Microsoft comprou o github por 7,5b.

Ativando verificação de duas etapas no github

1. Vai em settings
2. Password
3. Enable two-factor
4. Select a microsoft authenticator.

**Autenticação do TOKEN**

- único, revogável, limitado e aleatório.

- necessário para acessar repositórios privados;

1. Ir em settings no github
2. Developer settings
3. Personal acess tokens – tokens(classic) – genarete new token – generate classic
4. Configuracoes token
   1. Note – descricao
   2. Dia de expiração
   3. Escopo do token - repo
   4. Generate token

(atenção após sair da página o token some)

**Salvando credencial na máquina (token)**

1. Git config credencial.helper cache ou store – temporário ou permanete.
2. Git config –global credential.helper store
3. Git clone url – novamente ele irá pedir o nome do usuário e o token.

**Visualizar os tokens que já foram criados.**

1. Ir em settings
2. Developer settings
3. Personal acess tokens – token classic.

**Autenticação utilizando Chave SSH (Security Shell)**

1. Settings
2. Ssh and gpg Keys
3. Generating ssh Keys – documentation
4. Verificar se há chaves ssh – usando o git.
   1. Ls –al ~/.ssh – verificar no git se existe uma chave existente.
5. Gerar nova chave ssh
   1. Ssh-keygen –t ed25519 –c “your email.com” – usa o algoritmo ed25519 para criar a chave.
   2. Pede o local pra chave ficar (só dar enter) e em seguida pede uma frase de segurança.
6. eval “$(ssh-agent –s)” – executa o ssh agent, para adicionar a chave.
7. Ssh-add ~/.ssh/id\_ed25519 – adiciona a chave privada ao ssh-agent.
8. Agora adicionaremos a chave pública ao github.
9. Clicar em new ssh key
10. Adicionar title – key type: autentication – key: colocar a chave publica.
11. Para acessar a pagina: (digitar os commandos no git)
    1. Cd ~/.ssh
    2. Ls
    3. Cat id\_ed25519.pub
    4. Copiar a chave.

**Primeiros passos git e github**

1. Mkdir (cria uma pasta)
2. Cd (entra na pasta)
3. Git init (inicia um repositório) ( .git é uma pasta oculta.)
4. Git clone url (clona o repositório remoto) se colocar um nome após a url ele muda o nome da pasta que o git irá criar.
5. cat config – mostra a configuração do repositório.
   1. Mostra a origin e a branch
6. git remote –v (mostra quais são os repositórios vinculados)
7. git remote add origin url(repositório remoto) – no github clica em code e pega a url.
8. Cat config – irá mostrar apenas a informação do remote, já que ainda não teve nenhuma informação enviada pra branch.
9. Git clone url --branch feature-1 --single-branch (uma forma de clonar uma branch especifica)

**Salvando Alterações no repositório local**

1. Git init
2. Git status (mostra o status da arvore de trabalho, staging área)
3. Touch readme.md (cria um arquivo vazio)
4. Agora temos um untracked files, ainda não teve commit sobre esse arquivo.
5. Editando o readme.md
   1. Github docs- começar a escrever no github.
   2. Readme.so
      1. Para colocar link [c](url)
      2. Tabelas: |Aulas | Resumos| divisão |--------|--------|
6. Git add readme.md (insere um arquivo no nosso repositório)
7. Git commit –m “novo arquivo”
8. Git log (mostra os commits que foram feitos)
9. Atenção git não identifica pastas vazias. Pra reconhecer precisamos de arquivos dentro da pasta.
10. Echo resumos/ > .gitignore ( coloca a pasta resumos dentro do .gitignore)
11. .gitkeep (nas pastas vazias que você deseja que sejam reconhecidas, você coloca esse arquivo na pasta)
12. Git add . (adiciona todos os arquivos para commit)
13. Git commit –m “adicionar”
14. Git log (mostra os commits realizados.)

**Desfazendo alterações no repositório local**

1. rm –rf .git (Remover o versionamento da pasta )
2. git restore readme.md (restaurar arquivo da arvore de trabalho) esse comando elimina as últimas alterações no trabalho voltando com o arquivo anterior ao save. No caso o penúltimo save. Uma vez que você deu esse comando já era. Não tem volta;
3. git commit --amend -m “troquei a mensagem” ( esse comando muda a mensagem do último commit)
4. git commit --amend (altera diretamente digitando no terminal )
5. git reset (descarta o ultimo commit, retornando ao commit anterior.
   1. --soft
   2. --mixed
   3. –hard (apaga o commit e os arquivos)
6. Git reflog (histórico mais detalhado )

**Enviando para o repositório remoto**

1. Git remote add origin url
2. Git push –u origin main (set upstream )

**Baixar as alteracoes realizadas no repositorio remoto para o repositorio local.**

1. Git pull (puxa e mescla com o local)

**Editando no github**

1. Quando estamos logados, podemos usar a tecla “.” Para abrir o web editor no github. É igual um visual studio code online.

**Trabalhando com branches**

1. Git checkout –b teste (cria branch teste e passa a estar nessa) em questa de commit a main e a teste irão ambas esta apontando para o ultimo commit, antes do comando checkout.
2. Git add .
3. Git commit –m “branch teste criada”(cria um commit dentro da branch teste)
4. Agora o main ficou pra trás e seguimos na branch teste.
5. Se realizarmos um commit na branch teste. Ela só aparecerá nesta branch.
6. Git checkout main (voltamos para a branch main. Ela está no mesmo ponto que a deixamos, sem alteração.)
7. Git commit –v (mostra os commits de todas as branchs)
8. Git merge teste (merge as modificações na branch teste para a branch main, branch atual)
9. Git branch (lista as branchs)
10. Git branch --d teste (exclui a branch teste)

**Possíveis conflitos com o Git**

1. Conflitos de merge
   1. Quando acontece um conflito o git mostra que tem um conflito e modifica o arquivo com as duas modificações(conflito)
   2. A gente salva o arquivo correto
   3. Faz um commit e um push

**Git fetch (não faz o merge diretamente)**

1. Git fetch origin main (a gente baixa do repositório remoto, mas sem juntar com o repositório local)
2. Git diff main origin/main
3. Git merge origin/main

**Git clone de uma branch apenas**

1. Git clone url --branch teste --single-branch

**Deixando suspenso em um commit**

1. Git stash (arquiva modificacao pedente para commit)
2. Git stash list
3. Git stash pop (tras as alteracoes arquivadas)

**Criando um repositório**

[**https://Markdownguide.org**](https://Markdownguide.org)

Professor falvojr

1. Mkr pasta
2. Git bash
3. Git clone url
4. Git add .
5. Git commit –m “texto”
6. Git pull
7. Git push origin main

**O desafio**

[**https://drive.google.com/file/d/1IZu0qohv1JOmxjEra1lknDiiStU68bl4/view?usp=sharing**](https://drive.google.com/file/d/1IZu0qohv1JOmxjEra1lknDiiStU68bl4/view?usp=sharing)

1. Criar o repositório