# GitLab

Uitgelegd door twee bijen

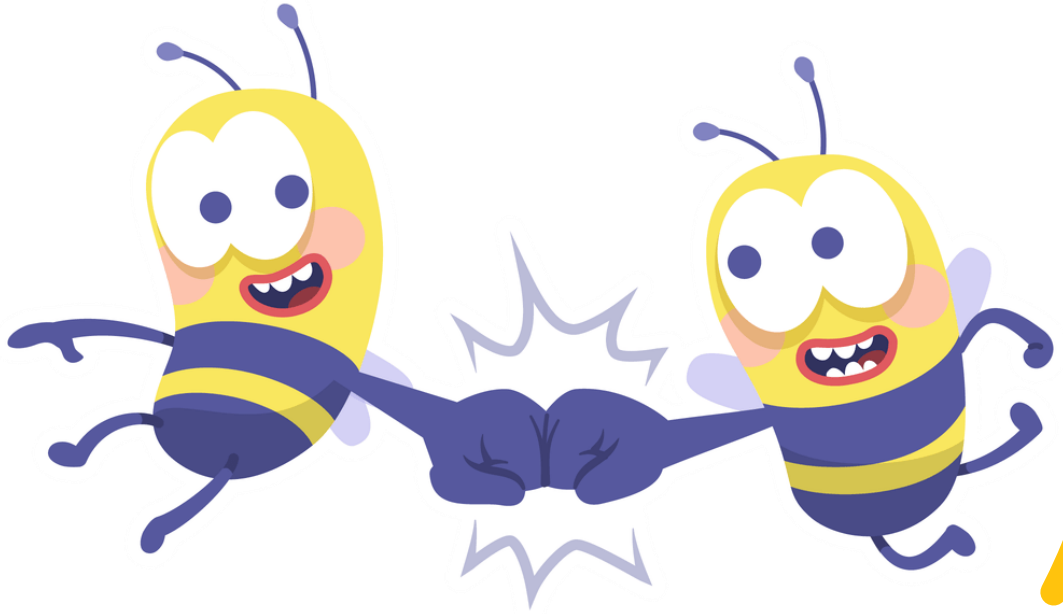
# Wat is Git?
# Waarom gebruiken we het?

- Via de cloud
- Versiebeheer
- Synchronisatie tussen verschillende computers/mensen
- Samen werken zonder elkaar in de weg te zitten
- Alles kunnen terugdraaien als je een spelbrekende bug hebt
- Overzicht en organisatie

# Standaardtermen

- Commit
  - Het klaar zetten van alles dat je op de git wilt hebben. Je kan nu nog kiezen of je het wel echt wilt pushen of niet.
  - Dit kan je ook gebruiken als je een merge nog wilt testen, voordat je het pusht (als de merge dan stuk is kan je het fixen voordat het gevolgen heeft).
- Push
  - Het versturen van je commit naar de server(cloud).
- Fetch
  - Ophalen of er veranderingen in de repository op de server(cloud) zijn.
- Pull
  - Het daadwerkelijk ophalen van wat er op de server(cloud) staat.

- Repository
  - Het project op GitLab



# Beginnen met een Git-project

- Ga naar GitLab
- Zoek het project op
- Clonen
  - Via https
  - Kopieer de link
- Open GitHub Desktop
  - https://desktop.github.com/
  - File > Clone Repository > URL
  - Fetch en pull

# Extra termen

- Commit message
  - Titel = animaties boombladeren
    - Korte samenvatting wat je gedaan hebt
  - Bericht = Animaties van de bladeren minder heftig gemaakt (na feedback)
    - Duidelijke uitleg en toelichting indien nodig
- Clonen
  - Het kopiëren/linken van het project naar je computer
- GitIgnore
  - Een bestand wat in je project-map zit wat ervoor zou moeten zorgen dat log-bestanden bijvoorbeeld niet steeds een nieuwe commit worden
    - (het werkt alleen vaak niet met Unity ☹)

# Branches

- Main-branche
  - No touchy
  - Alleen voor (tijdelijk) afgeronde producten, bijvoorbeeld per sprint
- Development branche
  - Huidig bug-free product

- Feature/…
  - Voor als je een onderdeel maakt
- Fix/…
  - Als je iets fixt, bijvoorbeeld een bug of een verandering van model
- Prototype/…
  - Om iets uit te testen wat eventueel een feature zou kunnen werken (bijvoorbeeld testen of een bepaalde code de game sneller maakt)

# Hoe maak je een branche aan?

- Open GitHub Desktop
- Ga naar de branche van waaruit je een nieuwe branche wilt maken
- Druk bovenin op deze branche-naam
- Druk op new branche
- Check of hij vanuit de goede branche wordt aangemaakt
- Bedenk een goed beschrijvende naam met correcte prefix (zie vorige slide)
- Maak hem aan en push hem

# Een goede workflow…

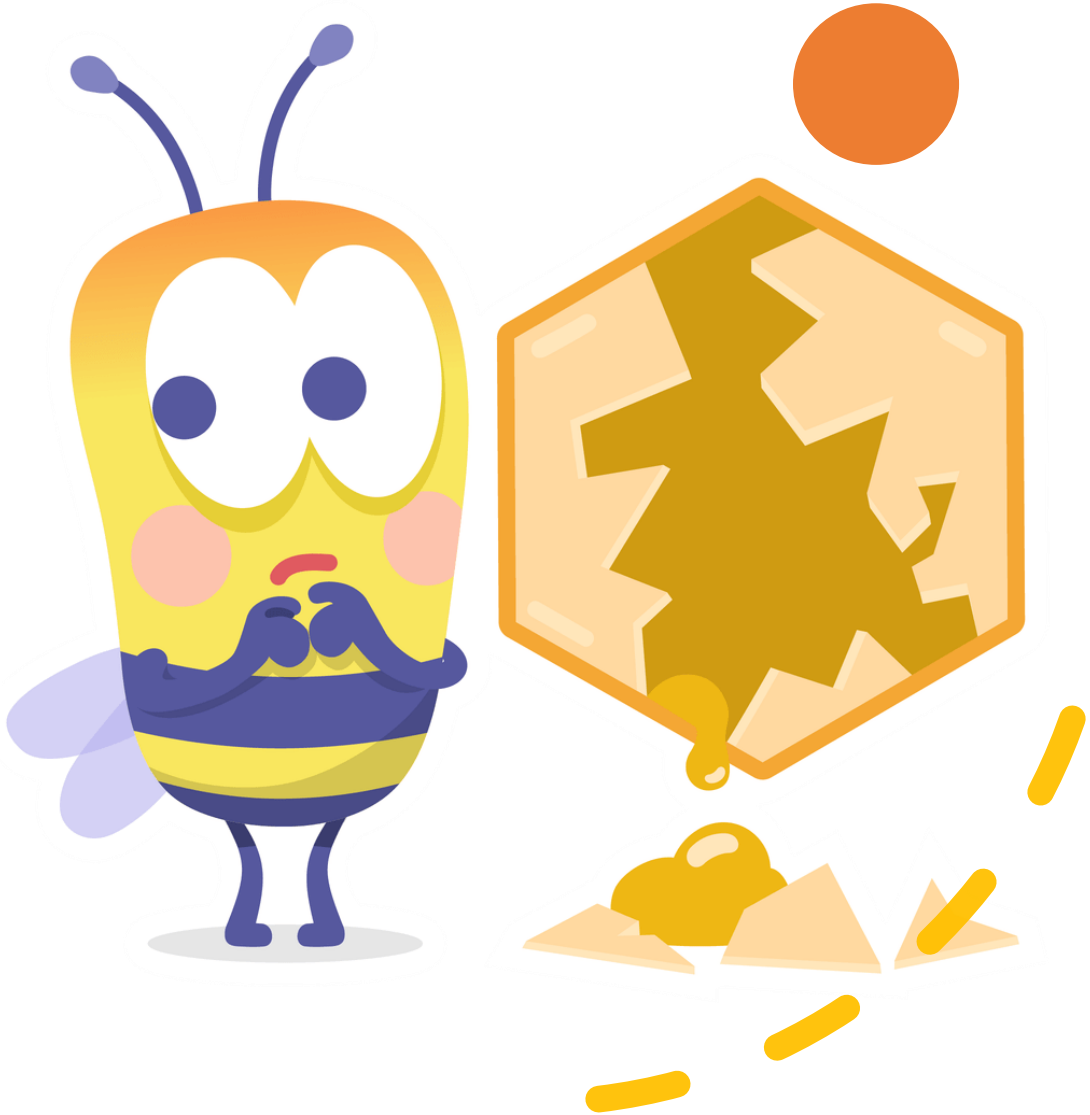
- Overleg als je bijvoorbeeld met een prefab of ander Unity onderdeel bezig gaat of niet iemand anders daar ook aan werkt om merge conflicten te voorkomen;
- Duidelijke commit berichten waarbij geen vragen nodig zijn;
- Heeft voldoende communicatie buiten Git! Overleg, houd elkaar op de hoogte…
- Werk in kleine hapbare stukken voor branches (dus liever tien feature branches dan één die te veel onderdelen heeft; ipv Main Character als één branche maak je er één voor zijn loopanimatie, één voor zijn shaders, etc)

# Merge requests

- Als je klaar bent met de huidige feature, etc
- Aanmaken via de website van GitLab
  - Ga naar de repository
  - Druk op merge requests
  - Maak er een aan met een titel zoals merge naamjouwbranche into naamanderebranche. Zet in de omschrijving wat je hebt gemaakt.
  - Klik aan dat tenminste één iemand moet reviewen om de merge goed te keuren.

# Merge conflicts

- Dit komt voor als twee branches in elkaar gemerged moeten worden, maar er een conflict is in hun inhoud;
- Voor nu het beste om dan gewoon een van ons erbij te roepen;
- Je moet dan per conflict kiezen welke branche de goede versie heeft.

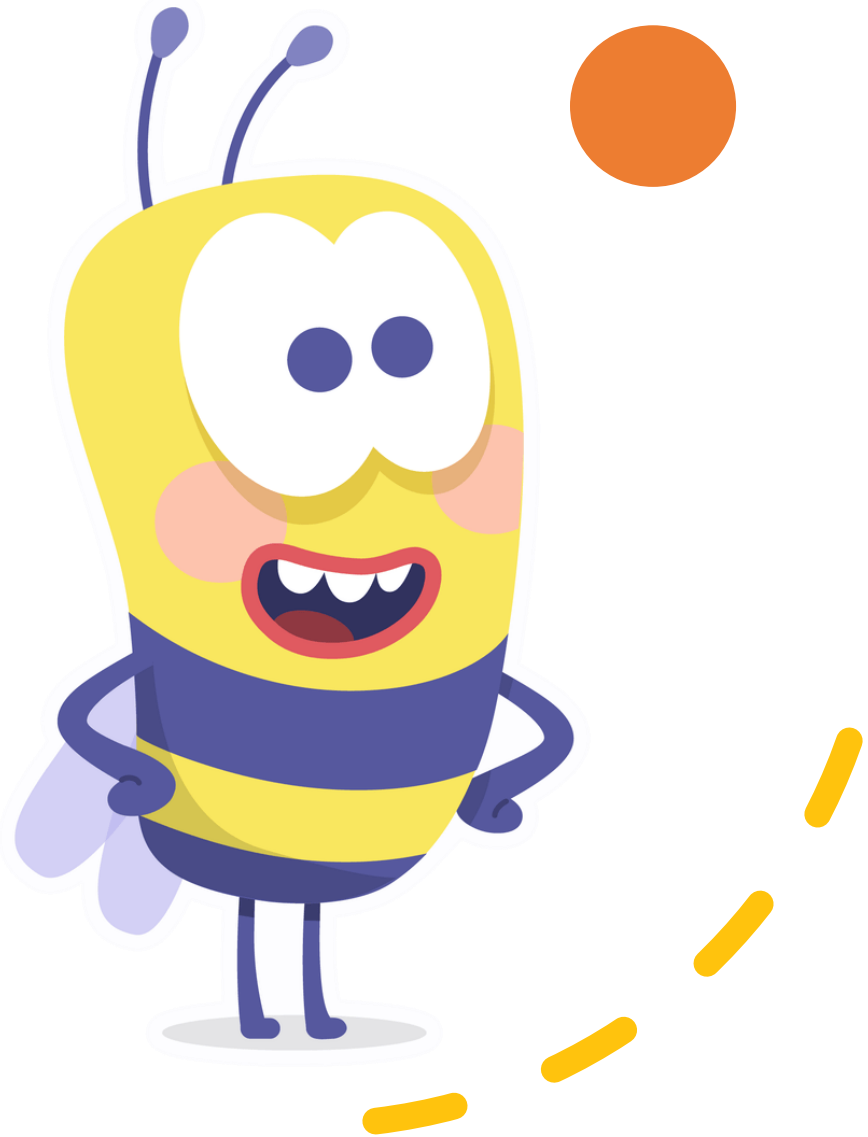


# Wat niet te doen

- In de branche van iemand anders gaan werken zonder overleg;
- Gefrustreerde commit messages die niks zeggen behalve dat niks werkt;
- Mergen naar development zonder:
  - development in je eigen branche te mergen;
  - alles bug-free te hebben;
  - te testen.
- Dingen naar main mergen zonder overleg;
- In de Unity-scene van een ander iemand werken;
- Je eigen merge-requests goedkeuren.

# Bonus: mappen structuur in Unity

- Animations
  - AnimatorControllers
- Audio
- Fonts
- Materials
- Models
  - Onderverdeeld in soorten, zoals characters/environment/props
- Prefabs
  - Zelfde onderverdeling als bij models
- Scenes
- Scripts
  - Onderverdeling op basis van samenwerking (bijvoorbeeld alles over besturing bij elkaar)
- Shaders
- Sprites (voor 2D games)
- Textures
- Third Party Assets
- VR



# Vragen?

# Aan de slag!