## はじめに

本演習の目的はチームで行うソフトウェア開発において利用されているバージョン管理システムが、どのような役割を担っているのかを実際にソフトウェアを開発することによって理解し使えるようになることである.また,その運用ルールをまとめたワークフローがどのような理由で提案されているのかを考える.

Githubアカウント e1b15030

## 準備

本演習では，分散バージョン管理システムの一つであるGitとGitrリポジトリのポスティングサービスであるGithubを利用する.また,Gitをチーム開発で利用する際の運用ルールをまとめたワークフローの一つであるGithub Flowにもとづいてチームでのソフトウェア開発を行う.開発においては，Processingと呼ばれる言語環境を利用し,「じゃんけんゲーム」と「どうぶつしょうぎ」の二つのゲームを作成する．

・バージョン管理システム

バージョン管理システム(版管理システム)は,ソースコードをはじめとしたファイルの履歴をバージョンとして保存・管理するためのシステムである.よって,誰が・いつ・どのファイルを・どのように編集したかを記憶することが可能で様々なメリットを享受できる．バージョン管理システムを利用せずにファイルを管理し，複数人でソフトウェア開発を行うとファイルを一箇所にまとめ共有することになる.このような状況では「どのファイルを誰がいつどんな目的で変更したのかわからない」,「ファイルを間違ってアップロードした場合簡単にやり直せない」,「変更箇所の意図しない重なりが生じる」などといった問題が発生し,その問題に対応する目的としてバージョン管理システムが利用されている.

・Github Flow

Github Flowとはチームソフトウェア開発のためのワークフローの一つである.チームソフトウェア開発では各メンバーが勝手に開発を進めたりするとソースコードが別の場所に紛れ込んだりと様々な問題が発生することが考えられるため,Github Flowでは六つの利用ルールを定めている．六つのルールではバグがmasterに入らないように二人以上で確認を行うようにしたり，ブランチの名前を第三者にわかりやすくすることなどによってチームソフトウェア開発がスムーズに行われるために活用されている．

## 課題

* 1. バージョン管理システムを利用することによるメリットを3つ以上述べよ．

・どのファイルがいつ誰がどのような目的で変更したかがわかる

・ファイルを間違えてアップロードした場合でも変更が容易

・編集作業の意図しない重なりを検出してくれる

* 1. Github Flowはチームでバージョン管理システムを用いて開発をするう上で発生する様々な問題を改善するためのルールを提供している.具体的にどのような問題が改善されるか,二つ以上の問題について述べよ.

・問題(1)

　ブランチ名に番号のみが振り分けられ第三者に何をしているのかが伝わらなかった.

・Github Flowによる改善

　Github Flowには「新しい何かに取り組む際は説明的でわかりやすい名前のブランチを作成する」というルールが定められていて,第三者に何をしているかが分かりやすいようになっている.

・問題(2)

masterブランチのソースコードにエラーが出て原因がわからない

・Github Flowによる改善

Github Flowには「masterブランチにあるものは何でもデプロイ可能である」というルールが定められていてmasterブランチには正常に動作するソースコードのみ存在するようになっており原因がわからないエラーが出た際はmasterブランチにある動作するソースコードからやり直すことが可能となる.

## Github Flowにもとづくじゃんけんゲーム開発

## 開発者情報

|  |  |
| --- | --- |
| 主開発者 | 共同開発者 |
| 氏名(自分)：金田優  学番(自分)：B15030  Github(自分):e1b15030 | 氏名：出口明日香  学番：B15067  Github:e1b15067 |

## リポジトリURL

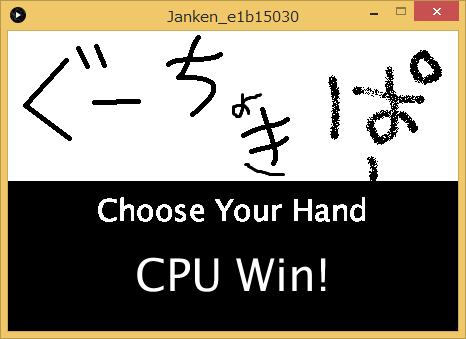
主開発者のGitリポジトリURLを下記URLに続けて記入せよ．

https://github.com/e1b15030/Janken\_e1b15030

## 開発状況

主開発者として開発したじゃんけんゲームの動作画面をキャプチャし，以下に貼り付けよ．また，正常系シナリオ(S0~D8)/異常系シナリオ(E1~E12)のどこまで完了できたかを記述せよ．

[じゃんけんゲーム動作画面]



[開発進捗状況]　 正常系シナリオD8まで完了

## ブランチ名の確認

主開発者として開発したじゃんけんゲームリポジトリにおいて，masterブランチ以外に作成されたすべてのブランチ名を以下に記述せよ．さらに，すべてのブランチ名が適切であるか（本来はそのブランチで何をするかが第3者に伝わる名前であるかを評価することが目的だが，ここでは演習中で指示されたものと一致しているかどうかを記述すること）．なお，ブランチ名は 2.で記述したGitリポジトリURL末尾に/branches/all をつけることでも表示できる．

ブランチ名「add\_janken\_display」

[ブランチ名に関する考察]

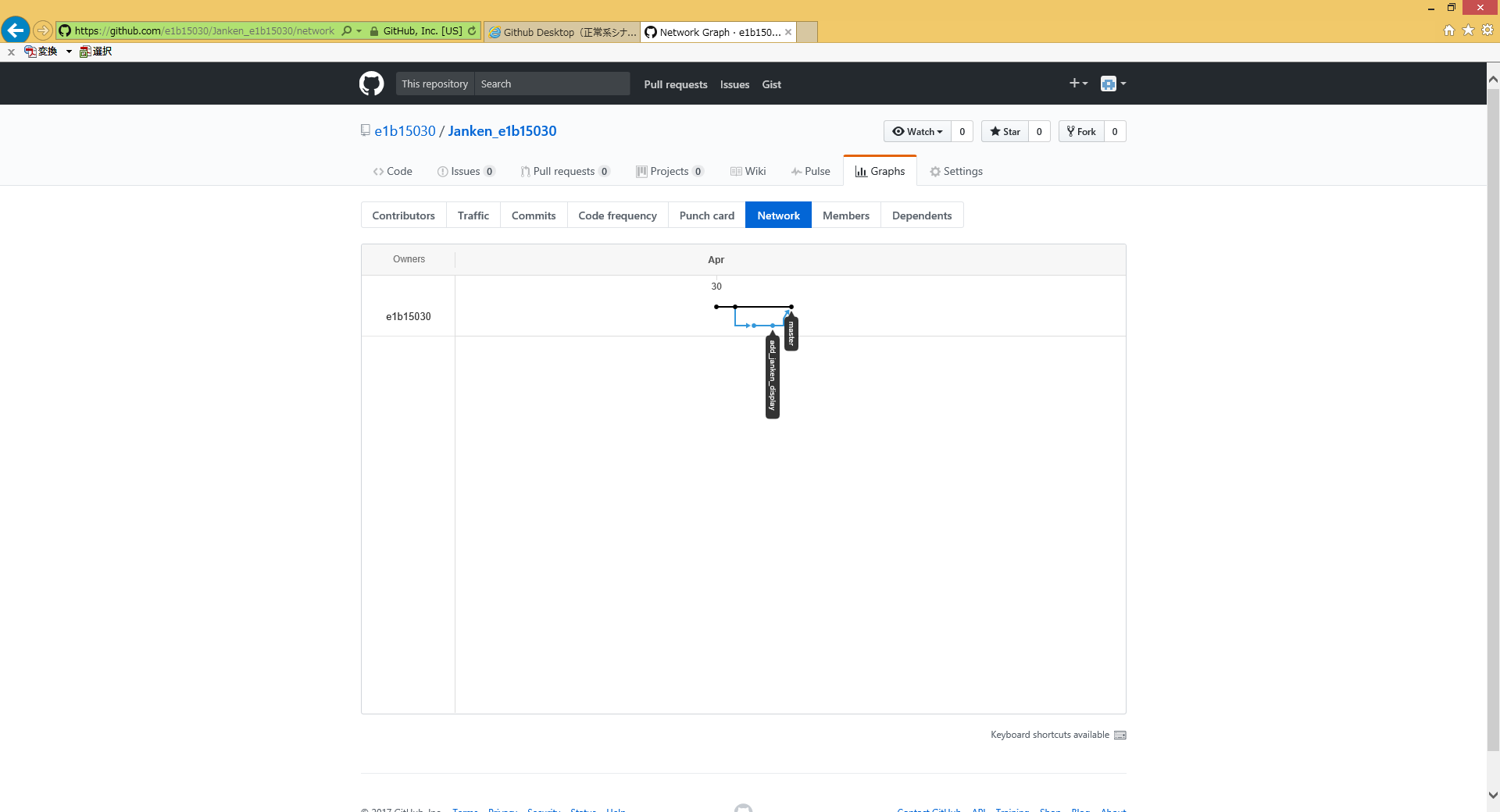
今回使用したブランチ名は演習中に指示されたものと一致しており適切であるといえる.また,演習中に指示された事を除いても比較的内容が第三者にわかりやすく名前の長さも長すぎないので適切であるといえる.

## コミットグラフ

主開発者として開発したじゃんけんゲームリポジトリにおいて， Githubサイトで表示されるコミットグラフをキャプチャし，以下に貼り付けよ．コミットグラフは，2. で記述したGitリポジトリURL末尾に"/network"をつけることで表示できる．  
その後，下記1~3それぞれについて対応できているかを記述せよ．

1. すべてのブランチがmasterブランチから作成されているか
2. すべてのブランチがMergeされているか
3. 初期コミット以外でmasterブランチに直接コミットされていないか（Pull RequestによるMergeならOK）

[コミットグラフ]



[ブランチ及びコミットに関する考察]

上のグラフは上記の1~3それぞれに対応している

## Pull Request

主開発者として開発したじゃんけんゲームリポジトリにおいて，Pull Requestの情報を下記に記述し，1,2に対応できているか確認し，記述せよ．Pull Requestは2. で記述したGitリポジトリURL末尾に"/pulls"をつけることで表示できる．Mergeが終了したPull RequestはClosedというリンクで，MergeされていないPull RequestはOpenというリンクをクリックすることで表示できる．

1. すべてのPRがMergeされているか
2. PR登録者とレビュアーが別の開発者になっているか

|  |  |  |  |
| --- | --- | --- | --- |
| PR URL(#) | タイトル | PR作成・登録者 | レビュアー |
| #1 | じゃんけん機能追加 | e1b15030 | e1b15067 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

ここで，PR URLにはGithubで割り当てられたPull Requestの#を記述すれば良い(#1,#2等)．PR作成・登録者及びレビュアーは担当した開発者のGithub アカウントを記述すること．

[Pull Requestに関する考察]

1. PullRequestは自分の変更したものを別の開発者にブランチで何が行われたのかを把握してもらったり,動作確認を行ってもらいバグ,エラーなどの早期発見のために使用されるのが好ましい.よってGithub Flowに基づいてチームソフトウェア開発を行うことによりPullRequestに対し他の開発者が確認しレビューをすることによりmasterブランチにバグなどが混入しにくくすることにつながっている.今回の開発ではすべてのPRがMergeされていて，PR登録者とレビュアーが別の開発者になっているので正しく使用できた．

## 4.7 じゃんけんゲーム開発実施中に発生したトラブルや得られた知見，開発における改善点

じゃんけんゲームの開発では，PRを一度しか行わなかったこともありエラーなどといったトラブルはみられなかった．しかし，最初に共同開発者との計画の話し合いが足りておらずお互いに質問が多く，時間を必要以上に使ってしまったのではじめに誰が，何を，どのように開発していくかを次回からは決めておくべきである．

## 5. Github Flowにもとづくどうぶつしょうぎ開発

どうぶつしょうぎの開発について，チームの行動を下記にしたがって振り返ること．なお，本シートはそのまま大レポートの一部となる．1チームに複数人の大レポート担当者がいた場合，個々の項目のデータについては共有して構わない．ただし，ブランチ名に関する考察やコミットグラフに関する考察等，考察内容については各自で記述すること．そこまでコピーしていた場合は両方共再提出を指示し，評価を減点する．

## 開発者情報

主開発者及び共同開発者の情報を下記に記述すること．また，自分の学生番号，氏名，Githubアカウントについてはアンダーライン(“\_\_\_”)を引いて分かるようにしておくこと．

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 主開発者 | 共同開発者1 | 共同開発者2 | 共同開発者3 |
| 学生番号 | B15067 | B15014 | B15030 | B15096 |
| 氏名 | 出口明日香 | 植山洋登 | 金田優 | 松本悠馬 |
| Github | e1b15067 | aiueo15014 | e1b15030 | e1b15096 |

## リポジトリURL

どうぶつしょうぎのGitリポジトリURLを下記URLに続けて記入せよ．

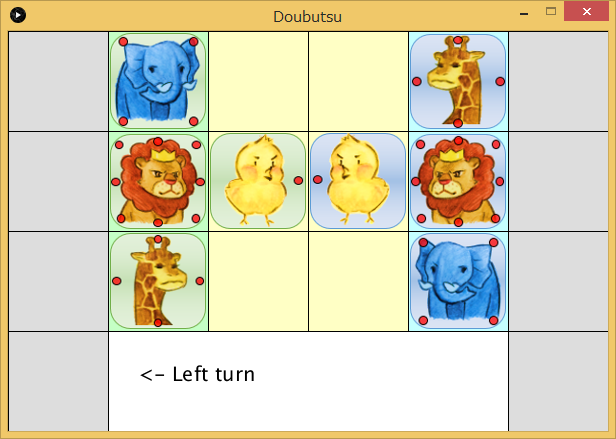
<https://github.com/e1b15067/Doubutsu>

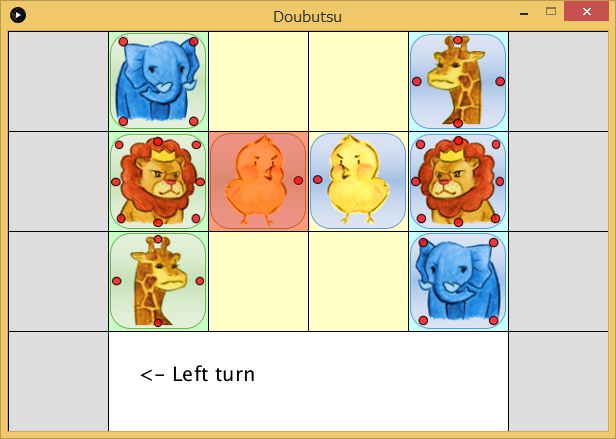
<https://github.com/e1b15067/Doubutsu2>

## 開発状況

どうぶつしょうぎの動作画面をキャプチャし，以下に貼り付けよ．また，全9Stepのどこまでできたかを記述せよ． Step内の特定のTaskまでの場合，どのTaskまでできたかもあわせて記述すること．

[どうぶつしょうぎ動作画面]





[開発進捗状況] Task3-5まで作成完了.

## 開発担当Task数とレビュー担当Task数の確認

チームメンバ毎の開発及びレビュー担当Taskを以下に記述せよ．さらに，それぞれの担当Task数を記述せよ．なお，開発・レビューのどちらか一つでも担当Task数が5に満たないメンバがいた場合は，開発のやり直しが発生することがあるので，チームメンバ及び教員に相談すること．最後に，開発・レビューTaskについて，下記の観点に基づいて考察せよ．

1. 特定のTaskについて，同じメンバが開発とレビュー両方を担当していないかどうか
2. 開発・レビューTask数がメンバ間で均等になっているか

開発担当，レビュー担当には各Task担当者のGithubアカウントを記述すること．

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 主開発者 | 共同開発者1 | 共同開発者2 | 共同開発者3 |
| 開発Task | Task0  Task1-3  Task2-2-c  Task2-2-d  Task3-4  Task3-5 | Task1-2-b  Task1-2-c  Task2-2-e  Task2-2-f  Task 3-3 | Task1-4  Task2-1  Task2-2-a  Task2-2-b  Task3-1 | Task1-1  Task1-2-a  Task2-3  Task2-4  Task3-2 |
| 開発 Task数 | 6 | 5 | 5 | 5 |
| レビューTask | Task 1-1  Task 1-4  Task 2-2-b  Task 2-3  Task 3-3 | Task 0  Task 1-2-a  Task 2-1  Task 2-2-a  Task 2-4  Task 3-4 | Task 0  Task 1-2-c  Task 2-2-c  Task 2-2-f  Task 3-2  Task 3-5 | Task 0  Task 1-2-b  Task 1-3  Task 2-2-a  Task 2-2-e  Task 3-1 |
| レビュー Task数 | 5 | 6 | 6 | 6 |

[開発・レビューに関する考察]

どうぶつしょうぎの開発において開発,レビューどちらもTask0以外被ることなく,バランスよく且つスムーズに振り分けること,取り組む事が出来た.

## ブランチ名の確認

チームで作成したすべてのブランチを下記に記述せよ．Githubのブランチリストをコピペするだけでも構わない． なお，ブランチ名は 2.で記述したGitリポジトリURL末尾に/branches/all をつけることで表示できる．

[チームで作成したブランチ一覧]

master, task1-1,Task1-2-b, Task1-2-c, Task1-3, Task1-4, Task2-1, Task2-2-a, Task2-2-b, Task2-2-c, Task2-2-e Task2-2-f, Task2-3.4,Task3-1, Task3-2, Task3-3,Task3-3-, Task3-3a, Task3-5

次に，ブランチ名を見るだけで，そのブランチでの開発内容が第3者でも理解できるものになっているかどうか，各ブランチ名について考察し，記述せよ．わかりにくい名前になっていると思ったものについては，良いブランチ名とあわせて記述すると良い．実際のブランチでの開発内容についてはコミットを見るか，チームメンバに聞いて確認すること．

[ブランチ名に関する考察]

ブランチ名については,班員内での意思の疎通がしっかりと出来ていなかったことで統一することができず,Githubが苦手で使いこなせていない班員に対してフォローができなかったことで同じようなブランチができてしまったり二つのTaskが一つのブランチに入ってしまいGithubu Flowのルールに従えずその結果Task3-5が完了した段階でエラーが出てしまい修正に時間がかかってしまった.

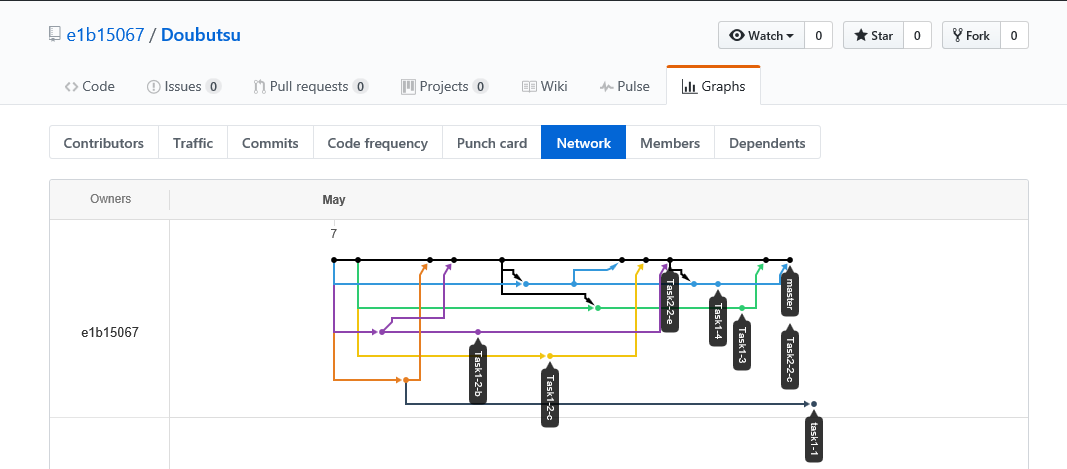
## コミットグラフ

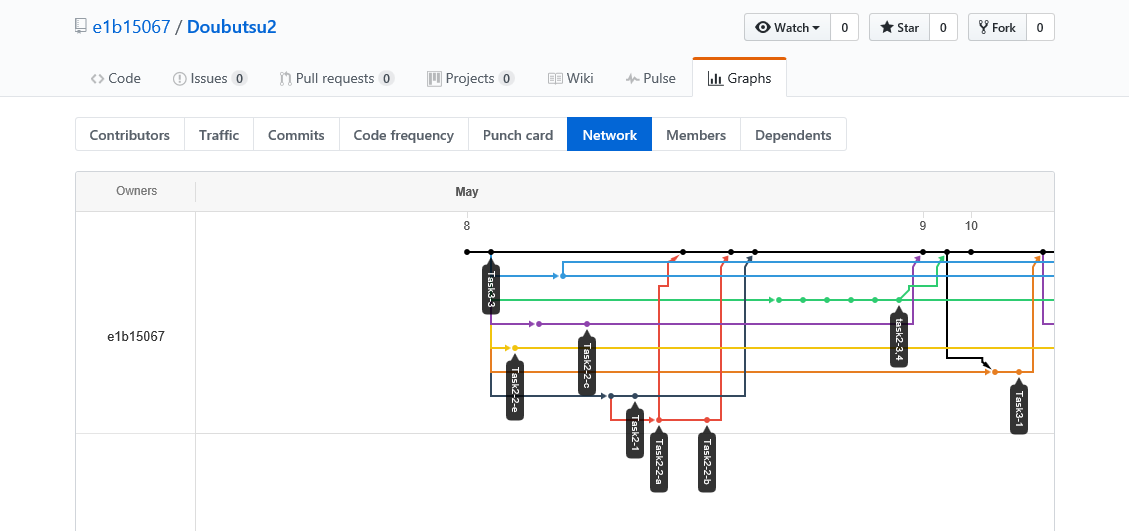
どうぶつしょうぎリポジトリにおいて， Githubサイトで表示されるコミットグラフをキャプチャし，以下に貼り付けよ．コミットグラフは，2. で記述したGitリポジトリURL末尾に"/network"をつけることで表示できる．その後，下記1~3それぞれについて対応できているかを記述せよ．

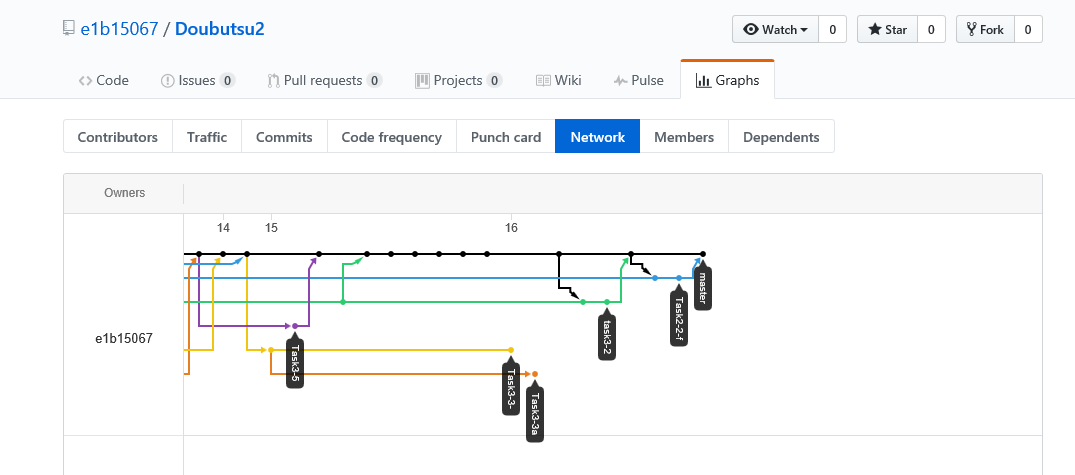
すべてのブランチがmasterブランチから作成されているか

1. すべてのブランチがMergeされているか
2. 初期コミット以外でmasterブランチに直接コミットされていないか（Pull RequestによるMergeならOK）

[コミットグラフ]







[ブランチ及びコミットに関する考察]

今回のチームソフトウェア開発では途中で不具合が生じたためTask2から作り直したためコミットグラフが２つ出来てしまった.また5.5の[ブランチ名に関する考察]で述べたように班全体でGithub Flowのルールを確認しあわなかったりGithubの使用が苦手な班員に対してのフォローが出来ていなかったため必要のないTask3-3a，Task3-3-などといった必要のないブランチが生成されたり,masterブランチで直接コミットが行われてしまった.

## Pull Request

すべてのPull Requestについて，下記に記述し，以下の1~3を守れているか考察すること．特に，3のPull Request作成・登録数が2に満たない場合は，開発のやり直しが発生する場合があるため，チームメンバ及び教員に連絡すること．Pull Requestは2. で記述したGitリポジトリURL末尾に"/pulls"をつけることで表示できる．Mergeが終了したPull RequestはClosedというリンクで，MergeされていないPull RequestはOpenというリンクをクリックすることで表示できる．

1. すべてのPRがMergeされているか
2. PR登録者とレビュアーが別のメンバになっているか
3. メンバごとのPull Request作成・登録数が2以上である

|  |  |  |  |
| --- | --- | --- | --- |
| PR URL(#) | タイトル | PR作成・登録者 | レビュアー |
| #1 | Task1-2-b MochigomaAreaの作成 | [aiueo15014](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Aaiueo15014) | [e1b15067](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15067) |
| #2 | AbとBe | [e1b15096](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15096) | [e1b15067](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15067) |
| #3 | Task1-2-b修正 | [aiueo15014](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Aaiueo15014) | [e1b15096](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15096) |
| #4 | Task1-2-c | [aiueo15014](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Aaiueo15014) | [e1b150](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15067)30 |
| #5 | Task1 4 | [e1b15030](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15030) | [e1b15067](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15067) |
| #6 | Task1 4 | [e1b15030](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15030) | [e1b15067](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15067) |
| #7 | Task1 3 | [e1b15067](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15067) | [e1b15096](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15096) |
| #8 | Task2-1AbstractKomaクラスとKomaStatusクラスの作成 | [e1b15030](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15030) | [aiueo15014](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Aaiueo15014) |
| #9 | AbstractKomaクラスとKomaStatusクラスの作成 | [e1b15030](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15030) | [e1b15096](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15096) |
| #10 | Task2 2 b | [e1b15030](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15030) | [e1b15067](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15067) |
| #11 | Task2 3 4 | [e1b15096](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15096) | [aiueo15014](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Aaiueo15014) |
| #12 | Task2 2 c | [e1b15067](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15067) | [aiueo15014](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Aaiueo15014) |
| #13 | Task3 1 | [e1b15030](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15030) | [e1b15096](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15096) |
| #14 | Niwatoriの作成 | [aiueo15014](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Aaiueo15014) | [e1b15096](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15096) |
| #15 | コマ画像データの登録 | [aiueo15014](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Aaiueo15014) | [e1b150](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15067)30 |
| #16 | Doubutsu.pdeクラスの更新 | [e1b15067](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15067) | aiueo15014 |
| #17 | Task3 2 | [e1b15096](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15096) | [e1b150](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15067)30 |
| #18 | Task3 2 | [e1b15067](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15067) | [e1b150](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15067)30 |
| #19 | Task2 2 f | [e1b15067](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15067) | [e1b150](https://github.com/e1b15067/Doubutsu/issues?q=is%3Apr+author%3Ae1b15067)30 |

ここで，PR URLにはGithubで割り当てられたPull Requestの#で良い(#1,#2等)．PR作成・登録者及びレビュアーはGithub アカウントを記述すること．

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 主開発者 | 共同開発者1 | 共同開発者2 | 共同開発者3 |
| PR作成数 | 5 | 5 | 6 | 3 |
| レビュー数 | 5 | 6 | 6 | 6 |

[Pull Requestに関する考察]

上の表よりPull Request登録者とレビュアーが別のメンバーで全てのPRがMergeできていて,メンバーごとのPull Request作成・登録件数が二つ以上を満たしていることがわかる.しかしレビューする際の確認がしっかりできていなかった事やレビューが行われる前にMergeをしてしまった班員もいてその結果エラーなどが多く確認された.このことによりGithub Flowに基づいたチームソフトウェア開発の重要性を確認することができた.

## 発生したトラブル及び得られた知見，開発における改善点

今回のチームソフトウェア開発ではエラーが多く見られた,これはGuthub Flowのルールを班内でしっかりと確認,徹底をしなかったことでルール通りに開発が行われなかったことが大きな要因だと考えられる.またGithubの使用が苦手な班員などに対して他の班員がフォローすることがしっかりとは出来ずその結果不具合も生まれた.これらのことより,今回の開発で得られた知見で最も大切なことは班員同士の話し合いである.ブランチの名前をどのように統一するか,誰が中心となって進めるかなど話し合い常に班員の動向を確認することが重要だと考えれる.

## 6. 考察

私たちの六班ではGithub Flowにもとづいたソフトウェア開発がしっかりと，実践できておらずその結果masterブランチにエラーの出るソースコードが混じってしまったり班員との連携が上手くとることができなかった.そして,その結果を受け止めGithub Flowの六つのルールを班員で確認し意識して取り組むと大幅にエラーの数を減らすことができGithub Flowなどのワークフローの必要性を理解することができた．

## 7.参考文献

情報システム基礎演習2017

## 8.おわりに

今回おこなったチームソフトウェア開発ではただ一人でソースコードを書いて進めるだけではなく一つの目標に対して複数人で同時進行していくものであった．そのためには同じ目標に取り組むメンバーなどに対して積極的に話し合いを行いどのように進めるべきかなどを，考えなければいけ無く初めての経験でありとても貴重な経験ができた．また今回上手くいかなかったものの改善点などを次回チームソフトウェア開発を行う際に生かしていきたいと思う．