## Github Flowにもとづくじゃんけんゲーム開発

今回の演習では，2つのじゃんけんゲームを開発した．そのうち，自分が主開発者となったものについて，共同開発者の分まで含めて振り返りを実施し，以下に記述すること．

## 開発者情報

|  |  |
| --- | --- |
| 主開発者 | 共同開発者 |
| 氏名(自分)：前田　剛志  学番(自分)：B16079  Github(自分):e1b16079 | 氏名：村山　慧  学番：B16094  Github:kmmk0924 |

## リポジトリURL

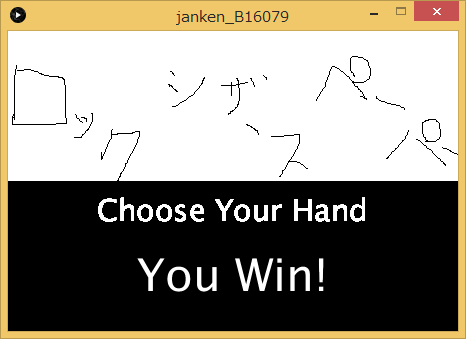
主開発者のGitリポジトリURLを下記URLに続けて記入せよ．

https://github.com/ e1b16079/janken\_B16079

## 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]

正常系シナリオの全工程を終了しました。

異常系シナリオは着手していません

## 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．

[add\_janken\_display](https://github.com/e1b16079/janken_B16079/tree/add_janken_display)

[ブランチ名に関する考察]

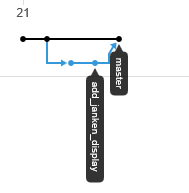
指示された名前と一致する。

## 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]

上記の３つの条件を満たす。

## 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 | e1b16079 | Kmmk0924 |

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]

上記の２つの条件を満たす。

## 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

Janken\_B16079.pdeを編集したがcommitとpushを忘れておりそのままPull Requestを送ってしまいそのままMergeしてしまった。まずPull Requestを送る前に確認するのを怠ったこと,そして確認不足のままMergeの許可を出したことが原因である。共同開発では複数人での確認を挟むことでミスを減らすことが目的である。その確認をおろそかにしたことが今回のミスの結果である。

## 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，自身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小レポートの一部となる．

## 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 | 共同開発者4 | 共同開発者5 |
| 学生番号 | B16031 | B16046 | B16094 | B16079 |  |  |
| 氏名 | 斎藤　巧 | 寺田　綾乃 | 村山　慧 | 前田　剛志 |  |  |
| Github | e1b16031 | piccolopico | kmmk0924 | e1b16079 |  |  |

## 5.2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/e1b16031/Doubutsu

## 5.3 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]

Step6：勝利条件チェックまでの工程を終了しました。

ライオンをとれば勝利となりますがひよこの昇格機能やコマの動きの制約などはありません

## 開発担当Task数とレビュー担当Task数の確認

自分が担当したTaskを開発，レビューそれぞれについて記述せよ．さらに，最後にそれぞれの担当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．

|  |  |
| --- | --- |
| 自分が開発を担当したTask | 自分がレビューを担当したTask |
| step1-1  step1-3  step3-3  step3-4  step3-5  開発担当Task数：5 | step3-1  step3-2  step4-1  step4-2  step4-3  レビュー担当Task数：5 |

## ブランチ名の確認

自分の作成したブランチ名とそのブランチでの開発内容を以下に記述せよ．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．

[自分の作成したブランチ名及びブランチでの開発内容一覧]

・add\_step1

AbstractAreaクラスの作成

Boardクラスの作成

・add\_step3-2

KomaListクラスの更新

Boardクラスの更新

Doubutsu.pdeの更新

・Doubutsu.health

step2のあとDoubutsu.pdeにエラーが出ていたのでその修正

次に，自分が作成したブランチ名について，第3者に開発内容が伝わる名前になっているかを考察し，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．

[ブランチ名に関する考察]

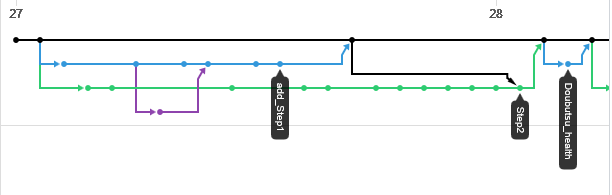
今回はstep3という名前にすると開発用のページがあったのでわかりやすかったが、これがないと途端に何をやっているかわからないものになる。このことからブランチ名は「step3」とするのではなく「コマ選択機能の追加」としたほうが良い。またstep3-2という名前で作ったがこれはstep3のtask2を表しているようでややこしくなってしまった。

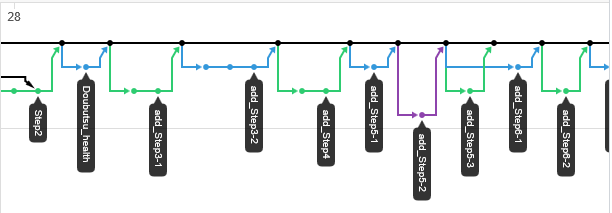
## コミットグラフ

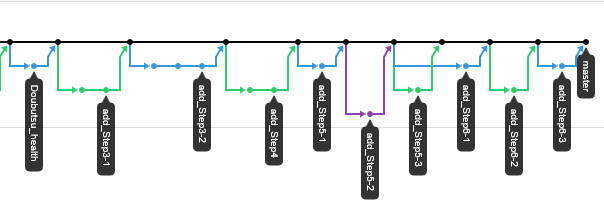
どうぶつしょうぎリポジトリにおいて， Githubサイトで表示されるコミットグラフをキャプチャし，以下に貼り付けよ．その後，下記1~3それぞれについて対応できているかを記述せよ．コミットグラフは，2. で記述したGitリポジトリURL末尾に"/network"をつけることで表示できる．これについては，自分が作ったブランチであるかどうかを考慮せず，すべてのコミットグラフを貼り付けて良い．

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]

上記の３つの条件を満たす。

## Pull Request

今回自分が作成・登録したPull Request及び自分がレビューを行った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開発者になっているか  
3.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 |
| ＃2 | Add\_step1 | e1b16079 | e1b16031 |
| #5 | 駒の選択機能の追加 | e1b16079 | e1b16031 |
| #4 | Add\_step3-1 | kmmk0924 | e1b16079 |
| #6 | Add\_step4 | e1b16031 | e1b16079 |

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．

[Pull Requestに関する考察]

上記の３つの条件を満たす。

## 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

はじめにstep1とstep2を同時に作成し同時進行で開発を進めていたら,step2がMergeできないということが起こった。これはstep1を開発しているときに作ったプログラムとstep2で作ったプログラムに違いがありこのまま上書きしてしまうのは危険である。とシステムがエラーを起こしていました。これは開発中のstep2に開発の終わったstep1をMergeして,step1とstep2の違いを消すということをした。先生には同時進行で開発するという方法もあるといわれたが,このトラブルから得た知見は前の作業が終わってないのに次の作業を始めるのは開発状況の混乱を招きやすいということがわかった。

8終わりに

今までプログラミングの演習は自分一人でやるものだったのでみんなにわかりやすく作るという作業が初めてで新鮮な演習だった。早く終わらせようとstep1とstep2を分担して同時進行で開発したがそれがトラブルにつながってしまった。急ぎすぎるのはダメなんだということと,チーム開発ではみんなの足並みをそろえ一つ一つ丁寧に作業を進めていくことが大事だとわかった。