



Formation NSI

lundi 5 décembre
Lycée Jacques Monod
Lescar

Amilcar DOS-SANTOS / Luc VINCENT

V 12.01

luc.vincent@ac-bordeaux.fr





Déroulement

• Matin

- Données structurées
 - réflexion et échange sur les notions (les listes, les tableaux, les dictionnaires ...)
- Parcours de listes et dictionnaires
 - index, élément, énumération
- Muable ou immuable

• Après midi

- Projet
 - Puissance4  Présentation
 - Light_out  Mise en Œuvre
 - Grille d'évaluation



Ressources

luc.vincent@ac-bordeaux.fr



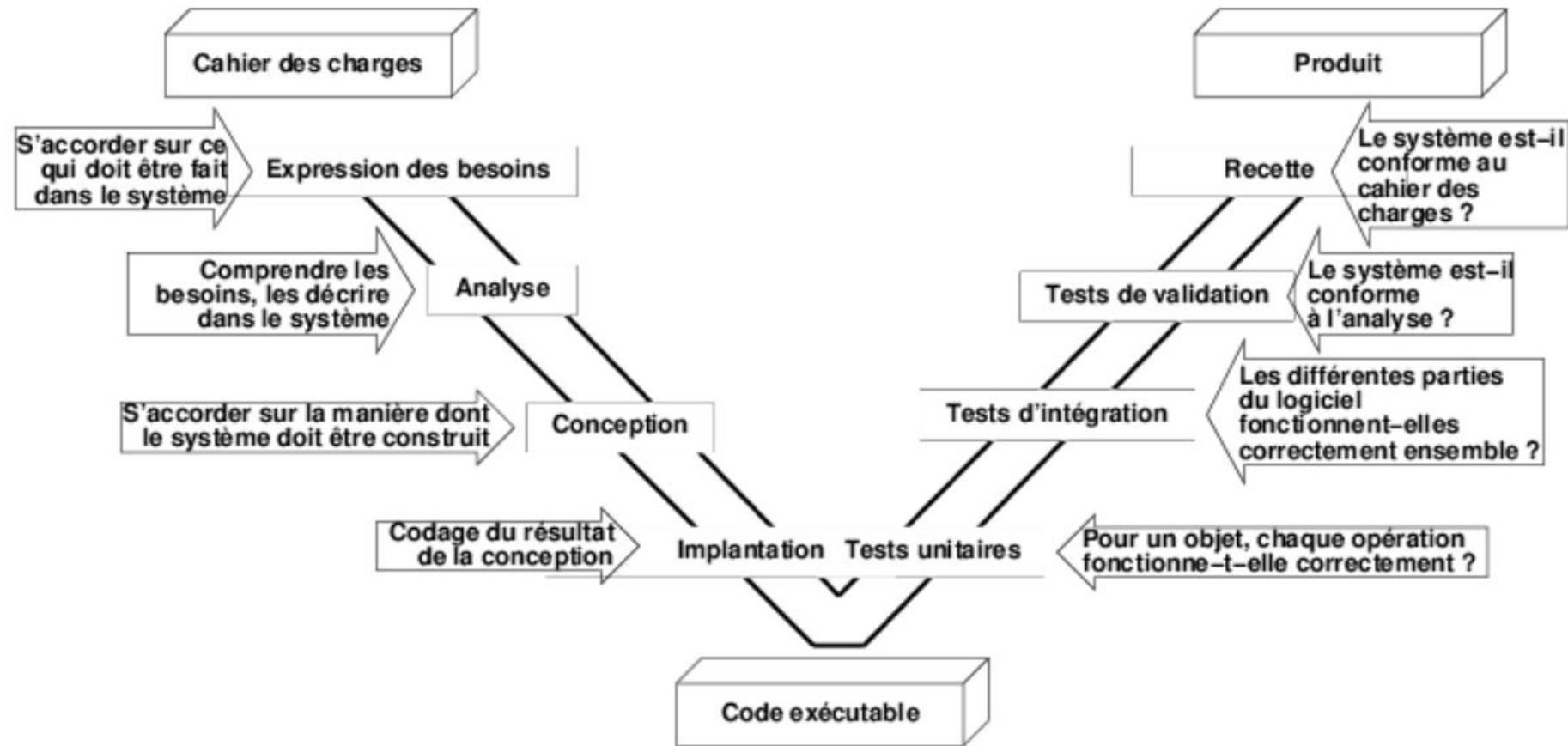
Construire un projet



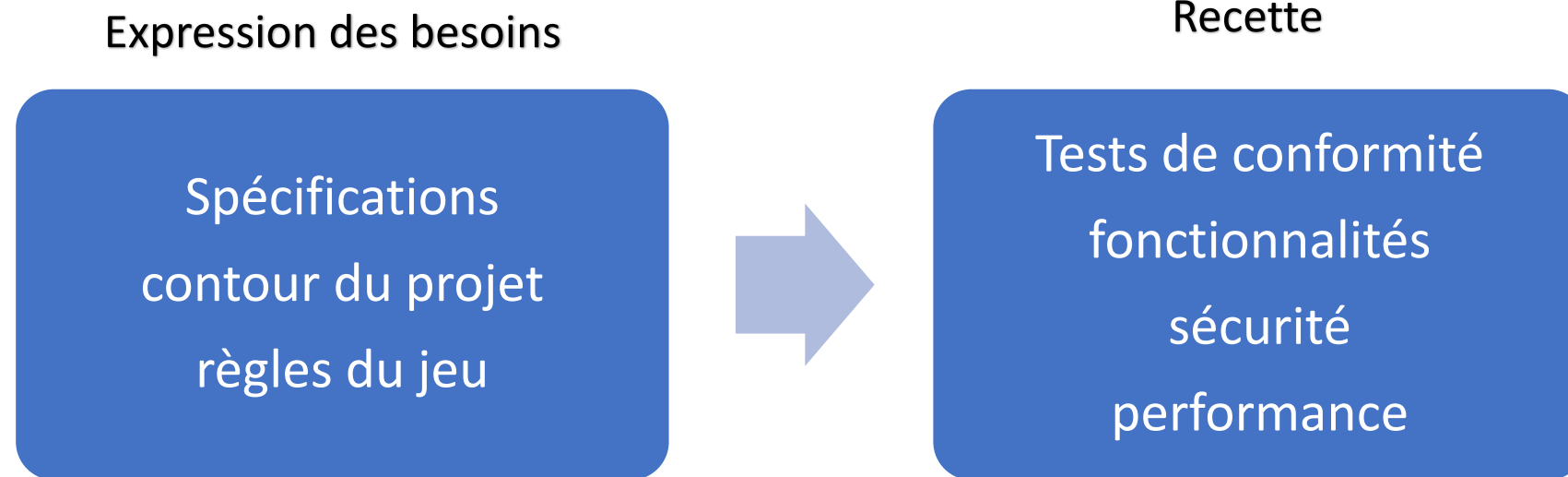
Conditions

- Chapitres déjà étudiés :
 - constructions élémentaires (affectation, condition, boucle, fonction)
 - tableaux
- Premier projet de première.
- Christophe.Viroulaud@ac-bordeaux.fr
- Présentation de la construction de projet à travers un exemple détaillé.

Cycle de vie d'un projet



Cycle de vie d'un projet

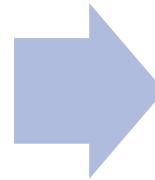


Cycle de vie d'un projet

Conception générale

Tests

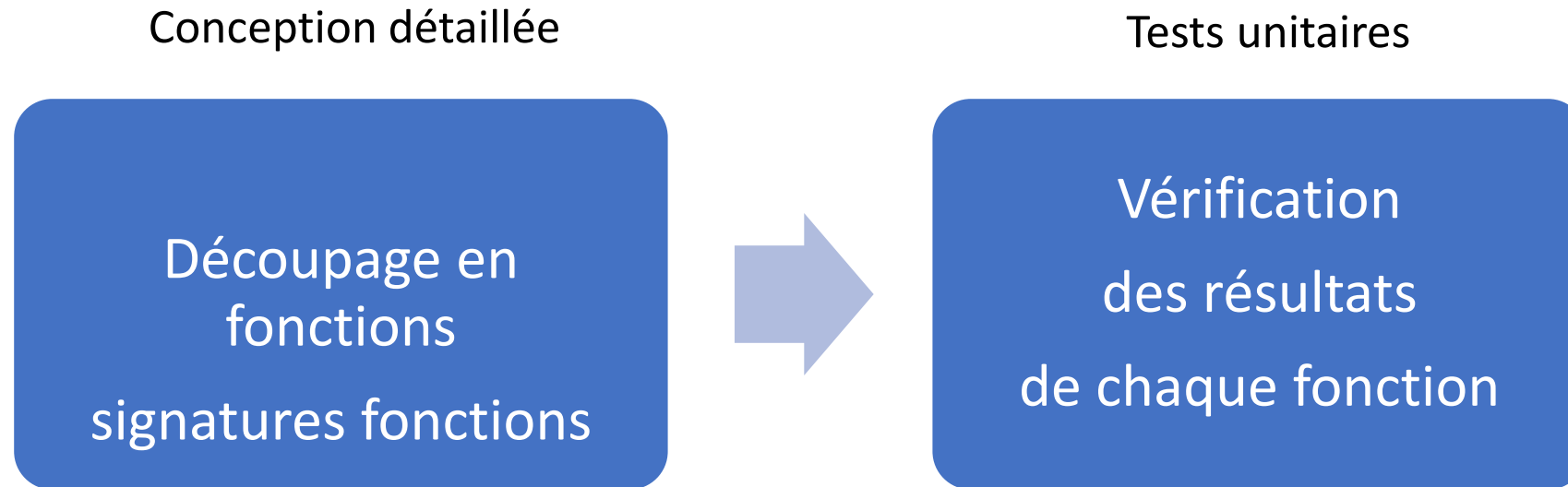
Conception générale
création des interfaces
déroulé du jeu



Tests d'intégration
bon comportement
des interfaces

Exemple : Respect des différentes séquences du jeu
(positionnement graphique du jeton...)

Cycle de vie d'un projet



Exemple : Vérification de chaque fonction de calcul du gagnant (vertical, horizontal)

Conception générale

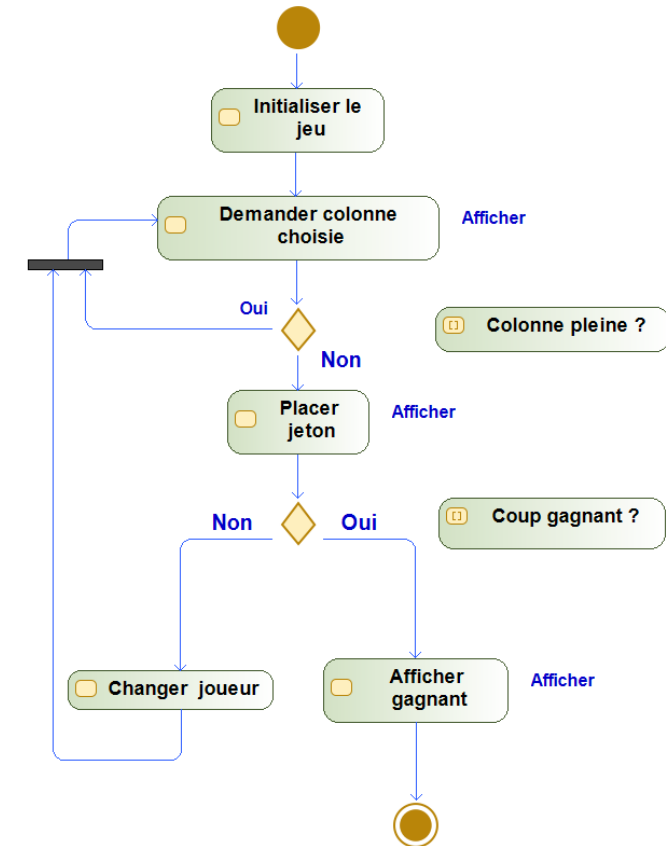
- Construire l'algorithme général du déroulé d'une partie.

Initialiser la grille

Tant qu'il n'y a **pas de gagnant** :

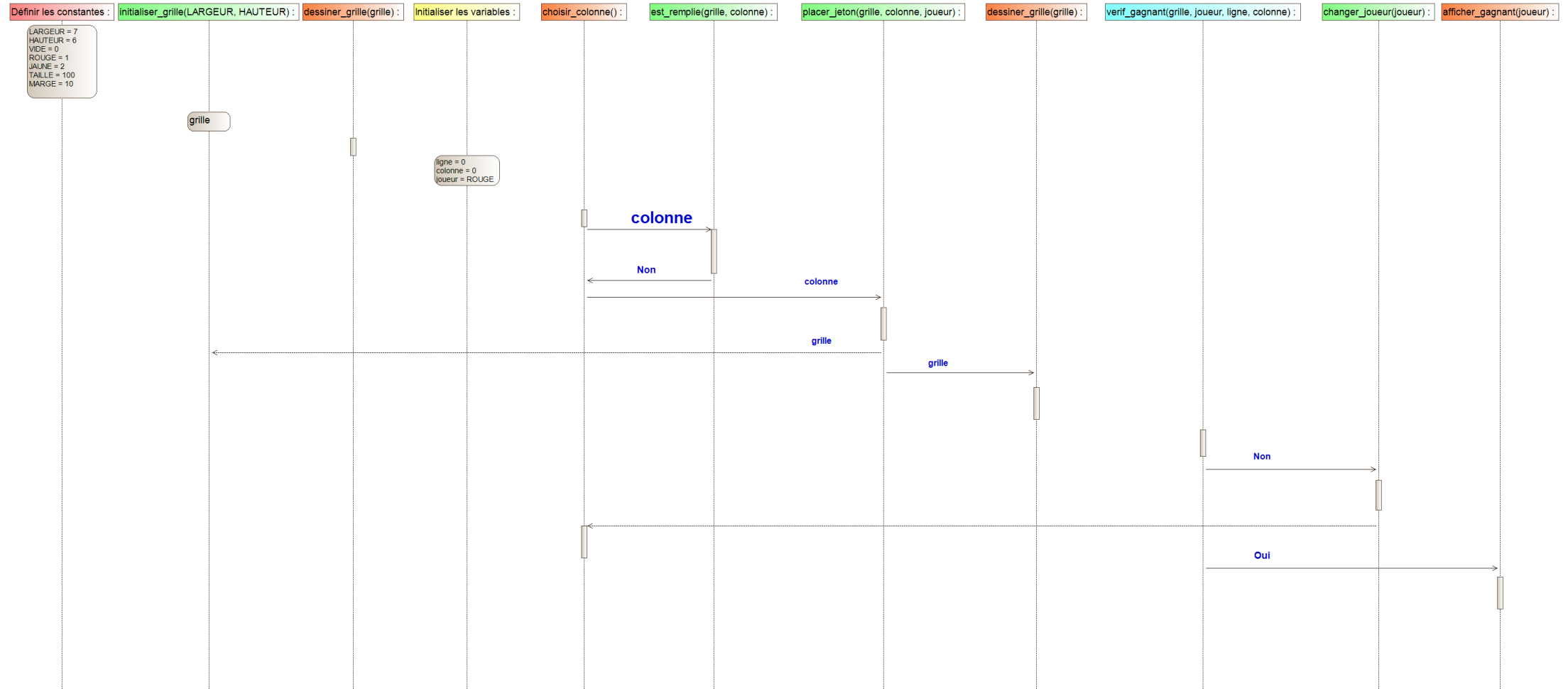
- ▶ **Choisir** le joueur suivant.
- ▶ **Demander la colonne** choisie.
- ▶ **Vérifier** que la colonne n'est pas **pleine**.
- ▶ **Placer** le jeton en le *laissant tomber* dans la colonne.
- ▶ **Afficher** la grille.

Partie terminée : **afficher** le gagnant.



Conception détaillée

• Activités



Conception détaillée

- Spécifier le rôle et la signature de chaque fonction.

`initialiser_grille`

- ▶ rôle : construire la grille du jeu
- ▶ paramètres :
 - ▶ `nb_col` : entier
 - ▶ `nb_lig` : entier
- ▶ renvoi : tableau de tableaux

`choisir_colonne`

- ▶ rôle : demande la colonne où poser le jeton
- ▶ paramètres : aucun
- ▶ renvoi : la colonne choisie

`est_remplie`

- ▶ rôle : vérifie si la colonne est remplie jusqu'en haut
- ▶ paramètres :
 - ▶ `grille` : tableau
 - ▶ `colonne` : entier
- ▶ renvoi : booléen, vrai si la colonne est remplie

`dessiner_grille`

- ▶ rôle : afficher la grille remplie
- ▶ paramètres :
 - ▶ `grille` : tableau
- ▶ renvoi : rien

`afficher_gagnant`

- ▶ rôle : afficher le nom du gagnant
- ▶ paramètres :
 - ▶ `joueur` : entier
- ▶ renvoi : rien

`placer_jeton`

- ▶ rôle : place le jeton dans la grille
- ▶ paramètres :
 - ▶ `grille` : tableau
 - ▶ `colonne` : entier
 - ▶ `joueur` : entier
- ▶ renvoi : entier, la ligne où le jeton est placé



Effet de bord sur la grille

`verifier_gagnant`

- ▶ rôle : vérifie si la partie est terminée
- ▶ paramètres :
 - ▶ `grille` : tableau
 - ▶ `joueur` : entier
 - ▶ `ligne` : entier
 - ▶ `colonne` : entier
- ▶ renvoi : booléen, vrai si le joueur a gagné

Conception détaillée

- Factoriser le code

Il sera peut-être nécessaire d'écrire d'autres fonctions pour exécuter certaines tâches *internes* :

- ▶ `verifier_horizontal`
- ▶ `verifier_vertical`
- ▶ `verifier_diagonal`

Conception détaillée

- Répartir le code en modules

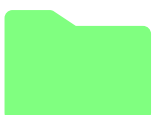


puissance4



constantes

```
LARGEUR = 7
HAUTEUR = 6
VIDE = 0
ROUGE = 1
JAUNE = 2
TAILLE = 100
MARGE = 10
```



fonctions_placement

```
choisir_colonne() -> int
changer_joueur(joueur: int) -> int
initialiser_grille(col: int, lig: int) -> list
est_remplie(grille: list, colonne: int) -> bool
placer_jeton(grille: list, col: int, joueur: int) -> int
```



rendu_console

```
dessiner_grille(grille: list) -> None
dessiner_jeton(grille: list, l: int, c: int) -> str
couleur_jeton(jeton: int) -> str
afficher_gagnant(joueur: int) -> None
```



fonctions_verif

```
verif_gagnant(grille: list, joueur: int, ligne: int, colonne: int) -> bool
verif_verticale(grille: list, joueur: int, ligne: int, colonne: int) -> bool
verif_horizontale(grille: list, joueur: int, ligne: int, colonne: int) -> bool
verif_diagonale_montante(grille: list, joueur: int, ligne: int, colonne: int) -> bool
verif_diagonale_descendante(grille: list, joueur: int, ligne: int, colonne: int) -> bool
```



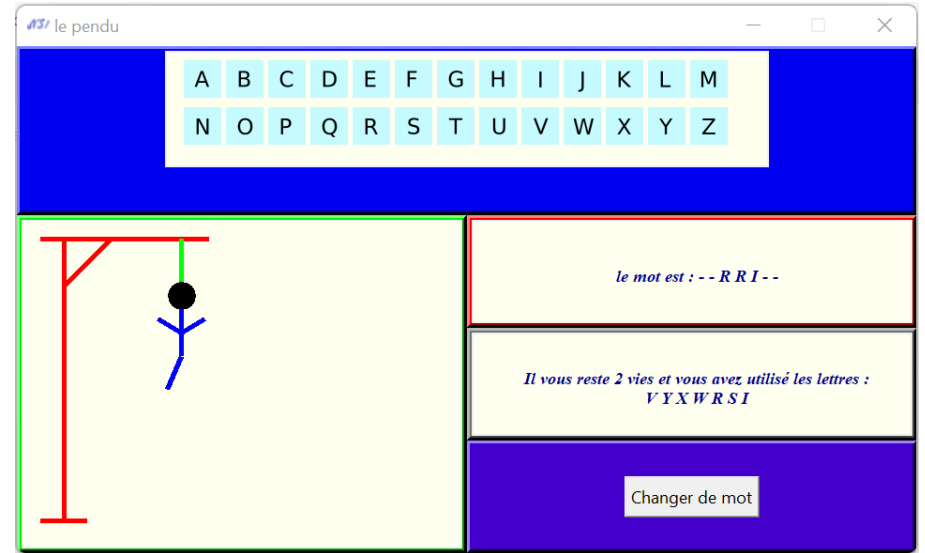
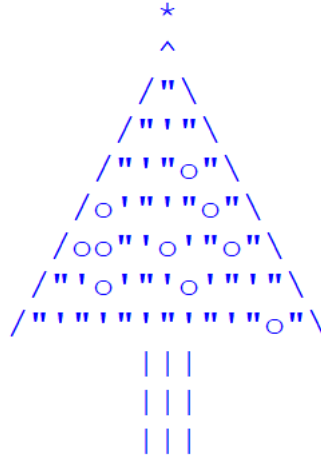
Fourni aux élèves

- Fournir le programme avec :
 - Le fichier de l'algorithme principal,
 - Toutes les signatures des fonctions,
 - Quelques fonctions non complétées.



Autres exemples de projets

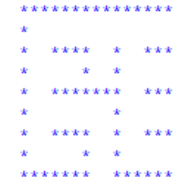
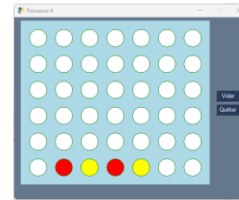
- 9_2_1 Sapin de Noël
- 9_2_2 La bataille
- 9_2_3 Le pendu (tkinter)



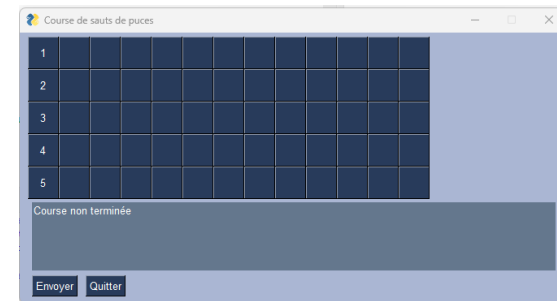
```
Partie 23 : Le joueur 1 a gagné la partie
Partie 24 : Le joueur 1 a gagné la partie
Partie 25 : C'est un match nul
Le paquet mélangé
[6, 2, 3, 2, 5, 4, 10, 8, 2, 10, 6, 1, 7, 4, 5, 3, 2, 8, 9, 9, 1, 9, 3, 7, 5, 9, 6, 7, 1, 10, 10, 4, 7, 4, 5, 8, 3, 8, 6, 1]
La main du joueur 1
[6, 3, 5, 10, 2, 6, 7, 5, 2, 9, 1, 3, 5, 6, 1, 10, 7, 5, 3, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
La main du joueur 2
[2, 2, 4, 8, 10, 1, 4, 3, 8, 9, 9, 7, 9, 7, 10, 4, 4, 8, 8, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
après la partie
Les cartes du joueur 1
[6, 3, 5, 10, 0, 6, 7, 5, 0, 0, 0, 0, 0, 0, 0, 0, 10, 7, 0, 0, 6, 2, 2, 4, 8, 0, 1, 4, 3, 0, 0, 0, 0, 0, 0, 4, 4, 0, 0, 1]
Les cartes du joueur 2
[0, 0, 0, 0, 10, 0, 0, 0, 8, 9, 9, 7, 9, 7, 10, 0, 0, 8, 8, 0, 0, 0, 0, 0, 2, 0, 0, 0, 2, 9, 1, 3, 5, 6, 1, 0, 0, 5, 3, 0]
C'est un match nul
```

Autres exemples de projets

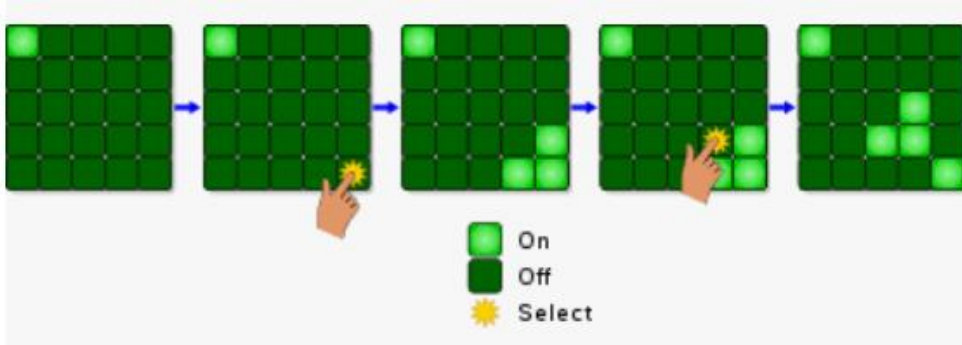
- 10.3 PySimpleGUI Puissance 4
- 10.4_22_NSIJ2ME1_5
- Nombre mystère en binaire xavier.dupin@ac-bordeaux.fr
- Jeu de briques marc.eldin@ac-bordeaux.fr
- Course de puce amilcar.Dos-Santos@ac-bordeaux.fr



D = Déplacer, Q = Quitter :



Light_out (Mise en Œuvre)



- Fiche de travail



Grille d'évaluation

- Concevoir
- Réaliser
- Communiquer



- Fiche de travail olivier.eloi@ac-bordeaux.fr

luc.vincent@ac-bordeaux.fr



	Indicateurs de performance
Concevoir	
Algorithme général	Le choix du modèle répond à la problématique à traiter.
Découpage et répartition des tâches	Le modèle choisi permet une répartition équitable et identifiable du travail à accomplir par chacun.
Reflexion sur la structure des données	Concordance du type des variables au sein du groupe
Conception détaillée	Signature et docString
Réaliser	
Documenter le code	Les commentaires apparaissent dans le code sur les parties difficiles à comprendre.
	Le choix du nom des variables est en adéquation avec le problème traité.
Respecter le modèle	Le code est divisé en fonctions ou objets conformément au modèle choisi.
Tester le code	Des tests sont intégrés au code
Respecter les licences	Le code emprunté est réutilisé dans le respect de la licence initiale
Communiquer	
Partager, suivre le projet	Les outils de recherche documentaire sont bien choisis et maîtrisés.
	Les outils numérique de partage sont utilisés
Présentation du projet	Grille du grand Oral