CodeTris

Figure 1 : Image représentant le projet

Alexandre King – MID4
Avenue de Valmont 28b, 1010 Lausanne
88h
Aurélie Curchod

Table des matières

1 RÉSUMÉ

Ce document permet de comprendre comment réaliser l'application CodeTris (Tetris combiné avec des questions de C#) en C# console ainsi que de reproduire celle-ci. Les outils utilisés au départ sont Visual Studio 2022 ainsi qu'un UWamp pour la BD.
Le but de CodeTris, est de pouvoir jouer à Tetris tout en complétant des questions sur le C#. Lorsqu'une ligne se bloque, une question est posée (donc si le joueur complète 4 ligne en même temps, 4 questions lui seront posés). Le public cible pourrait être des élèves de première année en informatique afin que ceux-ci puissent apprendre les bases de C# de manière ludique. Une planification initiale sera mise en œuvre au début du projet afin d'assurer au mieux le déroulement de celui-ci et d'arriver à le finir dans un temps imparti de 88 heures (documentation comprise).

Le projet sera réalisé en C# console avec comme template .Framework afin de profiter de certains aspects non-présent dans .NET. L'utilisation des classes permettra de bien segmenter le code afin de mieux suivre la logique du programme et le rendre plus « propre ». Certaines classes feront partie de la famille des managers. Elles seront les classes maitresses de certains points importants comme le jeu, le menu ou bien encore la base de données. D'autres classes feront, quant à elles, parties de la famille des classes statiques. L'utilisation de classes statiques permet de ne pas avoir les instanciées. Ces classes sont souvent appelées à beaucoup d'endroit dans le code ou non pas nécessairement besoin d'être instancié.
Les tetriminos seront gérés grâce à un tableau bidimensionnel représentant la grille de jeu (celle-ci étant plus grande visuellement, une grille avec les « vrais » dimensions est nécessaire). Pour les pièces en elles-mêmes, une classe Tetriminos sera parent de classes enfants pour chaque pièces (7 au total).
Les éléments seront abordés dans l'ordre suivant concernant la réalisation : Le menu, les options, la base de données, l'interface du jeu, la gestion des Tetriminos, les questions de C#, le game over, le score et enfin, la mise en base de données des parties. L'ordre suit la logique d'une partie lambda d'un joueur. Il commence le jeu dans le menu, (choisit les options pour sa partie,) lance le jeu, joue avec les Tetriminos, répond aux questions sur le C#, perd la partie, son score est alors enregistré à ce moment-là, le jeu lui affichage son score final et son nombre de bonnes et de mauvaise réponse, il peut alors retourner au menu principal.

//Résultats

Après 88 heures de projet, celui-ci est fini à XXX% (dire ce qu'il resterait à faire ou bien ce qui s'est bien passé, ce qui pourrait être amélioré)

2 SPÉCIFICATIONS

2.1 Titre

CodeTris : Le bloc de la programmation

2.2 Description

CodeTris est un jeu éducatif qui intègre des questions de programmation dans le jeu d'arcade Tetris.

L'objectif de ce projet est de créer une expérience ludique et éducative, encourageant les joueurs à renforcer leurs compétences en programmation tout en jouant à un jeu classique.

Ce jeu pourrait être utilisé par exemple comme accroche lors des modules de programmation de 1ère année.

2.3 Matériel et logiciels à disposition

- PC de l'ETML
- Visual Studio 2022
- Github
- Suite office

2.4 Cahier des charges

Voir annexes

2.5 Les points suivants seront évalués

- Le rapport
- Les planifications (initiale et détaillée)
- Le journal de travail
- Le code et les commentaires
- Les documentations de mise en œuvre et d'utilisation

2.6 Validation et conditions de réussite

- Compréhension du travail
- Possibilité de transmettre le travail à une personne extérieure pour le terminer, le corriger ou le compléter
- Etat de fonctionnement du produit livré

3 PLANIFICATION INITIALE

Cette partie montre la planification initiale du projet, elle montrera comment le temps a été réparti et quelles sont les paliers critiques. Elle indiquera également le début et la fin du projet ainsi que les jours fériés.

3.1 Répartition du temps

Cette partie montre la répartition du temps en heure et en pourcent ainsi que la répartition selon le cahier des charges afin de comparer.

Partie du projet concerné	Temps planifié (En heure et en pourcent)	Temps recommandé selon le CDC
Analyse	5.42 heures 6.15%	17.6 heures soit 20%
Réalisation	43.33 heures soit 49.23%	39.6 heures soit 45%
Documentation	35.26 heures soit 40.06 %	22 heures soit 25%
Tests	5 heures soit 5.68 %	8.8 heures soit 10%

3.2 Dates importantes

Le projet a débuté le 29 avril 2024 et il se finit le 29 mai 2024 (pour des raisons d'absences lors du projet, la date de fin a été déplacé au 30 mai 2024).

Les jours suivant ne sont comptés lors de la réalisation du projet :

- Tous les mardis et tous les jeudis matin (cours de MATU)
- Jeudi 9 mai et vendredi 10 mai (Ascension)
- Lundi 20 mai (Pentecôte)

3.3 Planification dans Excel

Pour des raisons de mise en page et de lisibilité, le diagramme de Gantt fait dans Excel se trouve dans les annexes.

3.4 Méthode de gestion de projet utilisée

La méthode de gestion de projet utilisée est la méthode des 6 pas.
Elle consiste en :

1. Informer (s'informer sur le projet)
2. Planifier (planifier en fonction du temps et des moyens/demandes du client)
3. Décider (choisir comment sera fait le projet (sélection du template))
4. Réaliser (réaliser le projet)
5. Contrôler (vérifier son bon fonctionnement avec des tests)
6. Évaluer (évaluer le résultat (fait par les experts et la cheffe de projet))

3.5 Modification de la planification

Comme précisé au point 3.2, dû à une absence, le délai a été repoussé au jeudi 30 mai. Il faut donc compter un décalage d'une demi-journée à partir du mercredi 1 mai.

4 ANALYSE

4.1 Opportunités

- Mise en œuvre d'un jeu avec C# console
- Utilisation de GitHub
- Jeu éducatif pour des élèves de première année

4.2 Objectif

Le but du projet est de réaliser un jeu mêlant tetris et le C#. Il pourrait être présenté à des élèves de première année dans le but de leur faire apprendre le C# de manière ludique.

4.3 Document d'analyse et conception

4.3.1 Analyse de l'application

Une analyse de l'application serait faite dans un premier temps. Ensuite, la conception sera abordée au point 4.3.2

Chaque élément important aura droit à sa section lors de cette analyse.

4.3.1.1 Général

Les éléments principaux ne faisant pas partis d'une section spécifique seront abordés dans ce chapitre.

Le template utilisé sera C# console .Framework. Dans le cas où du son serait ajouté plus tard dans l'application (ajout venant du point 4.3.1.5), il sera nécessaire de pouvoir utiliser le using System.Media permettant d'utiliser les SoundPlayer. Ceux-ci n'étant pas disponible avec C# console .NET, il faut passer par un moyen bien plus long pour pouvoir ajouter du son.

4.3.1.2 Menus

Le menu principal doit contenir les éléments suivants :

Nom du choix	Description
Jouer	Permet de lancer le jeu
Options	Affiche le menu des options
Meilleurs scores	Permet au joueur de voir les meilleurs scores en fonction de la difficulté
Tuto / aide pour le joueur	Tuto expliquant en vitesse le jeu (doc utilisateur intégrée)
Quitter	Quitte l'application

Le menu des options doit contenir les éléments suivants :

Nom du choix	Description
Touches	Permet de choisir si l'on veut jouer avec les touches WASD ou les flèches

Difficulté	Change la difficulté (elles proviendront de la base de données)
Retour	Revient au menu principal

Ayant déjà prévu d'ajouter du son, une option « musique » sera également ajoutée. Elle permettra de couper ou lancer la musique.

Concernant le menu des meilleurs scores, celui-ci sera sous forme de sous-menu avec où l'on pourra choisir la difficulté pour laquelle on souhaite voir les scores. Un score est représenté par un nom de joueur (pseudo), un nombre de points et la date du record.

Le déplacement dans le menu se fera via les flèches. Ce symbole « > » permettra de voir quel sous-menu/option est sélectionnée et la touche ENTER permet d'afficher notre sélection.
Dans le menu des options, le mot « musique » passera de rouge à vert selon si elle est activée ou non. Pour les touches, il sera écrit à côté du texte « touches : » quelles sont les touches actuellement actives (« WASD » ou « Flèches »). Concernant la difficulté, il sera écrit à côté du texte « difficulté : » quelle est la difficulté actuelle. De plus, une couleur sera attribuée à chacune d'entre elles (vert = facile, jaune foncé = moyen, rouge = difficile).

Le menu principal bénéficiera également d'éléments visuels ''décoratifs'' comme un titre en ascii et des Tetriminos géants afin de directement reconnaitre le jeu au lancement.

4.3.1.3 Jeu

Un tetris fera office de base du jeu. Il faudra donc y intégrer les éléments suivants :

- Une grille où les pièces tomberont
- Des tetriminos (ceux du jeu de base avec leur couleur)
- La possibilité de faire pivoter les tetriminos
- La possibilité de les déplacer sans qu'ils ne sortent de la zone dédiée
- Ils devront s'empiler

Lorsqu'une ligne sera complétée, une question sur le C# sera posée. Le nombre de question correspond au nombre de ligne pleine (Exemple, si le joueur rempli 1 ligne, alors 1 question lui est posée. S'il rempli 3 lignes, 3 questions lui sont posées. Le nombre max de question est donc de 4 car au maximum, seulement 4 lignes peuvent être complétés à la fois).
Les questions seront stockées en base de données et une difficulté leur sera attribuée. Plus une question est difficile, plus elle rapporte de points. Si le joueur répond correctement à la question, des points lui sont attribués, en revanche, s'il répond mal, aucun point ne lui seront attribué et il subira en plus un malus. Pour l'instant, le seul malus demandé est de bloquer une ligne, réduisant ainsi la surface de jeu. En cas de mauvaise réponse, la réponse attendue s'affiche sur l'écran et un signal visuel averti le joueur. Lorsque le joueur atteint le haut du niveau avec un tetriminos, la partie s'arrête déclenchant le ''game over''. Le joueur voit alors son score ainsi que son nombre de réponses correctes et incorrectes. Il peut ensuite retourner au menu principal pour relancer une partie.

4.3.1.4 Base de données

La base de données devra stocker les éléments suivants :
- Les utilisateurs
- Les difficultés
- Les questions
- Les parties des utilisateurs

On fera d'abord un MCD afin de conceptualiser la base de données. Le MCD et MLD seront abordés dans la conception
//mettre le point conception DB
Dû au fait qu'une base de données sera présente, il faut également penser à ajouter un fichier de logs en cas de problème avec celle-ci. Pour éviter que le programme ne crash en cas de problème avec la base de données, un try catch sera mis en place. Le catch récupérera l'erreur et l'inscrira dans le fichier de logs. Il faudra donc gérer des documents extérieurs au programme grâce au StreamWriter et StreamReader et au using System.IO.

4.3.1.5 Améliorations possibles

Il est déjà possible d'imaginer plusieurs améliorations au programme. En voici quelques-uns :
- Ajout de musique dans les menus et le jeu
- Possibilité de choisir son pseudo ou de le modifié
- En jeu, une zone dédiée montre le prochain tetriminos
- Ajout de difficulté en plus
- Ajout de malus en cas de mauvaise réponse à une question
- Un menu « pause » lorsque l'on est en jeu

4.3.2 Conception de l'application

4.3.2.1 Menus

Chaque option du menu sera affichée comme une liste que l'utilisateur pourra parcourir avec les flèches. Ce signe ''>'' montrera sur quelle option l'utilisateur se trouve actuellement. Il validera le choix avec la touche ENTER. Pour les sous-menus, une option pour revenir en arrière sera présente. Pour gérer les choix des menus, l'utilisation d'un dictionnaire est pertinente afin d'avoir une clé unique pour les choix associé à un string (Dictionnaire avec un int et un string).

4.3.2.2 Jeu

4.3.2.3 Base de données

4.3.2.4 Autre ?

Jeu (tableau 2 dimension pour la grille tetris)
Gestion de la rotation,
Pas de thread (aide pour l'optimisation)
//Faire des chapitres comme pour 4.3.1 et montrer les maquettes et schémas ici
//Pour DB, faire tableau avec champ, taille, type, description, etc…

- Ce paragraphe décrit le fonctionnement de manière détaillée.
  - Autant que possible de manière graphique, imagée, tableaux, etc.
  - Tous les cas particuliers devraient y être spécifiés…
- Il s'agit d'y présenter les fonctionnalités à développer :

- Découpage en étapes, en modules, en fonctionnalités, etc.
- Schémas de navigation, schémas événementiels, structogramme, pseudocode, etc.

➢ Si le projet inclut une base de données :
- Dictionnaire des données
- Modèle conceptuel des données, modèles logique des données.

4.4 Conception des tests

Tableau des tests à réaliser lors du projet. Les résultats seront inscrits dans la partie test de ce document.

Nom du Test	Fonctionnalité testée	Description	Condition de réussite	Importance (0 = bas, 5 = haut)
Menu	Menu	Le joueur peut se déplacer dans le menu ainsi que quitter le jeu via le menu principal	Le joueur peut accéder à toutes les parties du menu et des sous-menus	4
Options	Menu d'options	Différentes options peuvent être sélectionnées. Celles-ci impacteront le jeu	Les options en jeu correspondent bien à celles choisies par l'utilisateur	3
Pseudo	Pseudo du joueur	Le joueur possède un pseudo unique. Il peut le définir lors du lancement du jeu	Le pseudo défini par le joueur est unique. Lors du premier lancement du jeu, il le défini	3
Déplacement des pièces	Déplacement et pivotement des pièces	Le joueur doit pouvoir déplacer les pièces horizontalement, les faire pivoter, et accélérer leur descente	Le joueur peut utiliser les touches choisies dans le menu « options » pour effectuer les différentes actions	5
Questions	Questions	Des questions apparaissent lors de la complétion d'une ligne	Les questions s'affichent et le joueur peux y répondre. Elles proviennent de la DB	4
Score	Changement du score	Incrémentation du score lors d'une bonne réponse à une question	Si le joueur répond correctement à une question, les score augmente en fonction de la difficulté	2
Game over	Activation au bon moment	Lorsqu'une pièce a atteint le haut de la zone, la partie est terminée	Le game over s'active uniquement lorsqu'une pièce touche le haut de la zone. La partie est ensuite	3

			sauvegardée en DB	
DB	Bon fonctionnement de la DB avec le jeu	La connexion avec la DB est faite correctement. Des informations doivent pouvoir être récupérées et insérées	Les informations sont correctement récupérées et insérées. Un fichier de log est prévu en cas de problème	4

Les tests seront faits au fil du temps et dans un ordre logique (certains tests ne pouvant être fait avant d'autres dû à un ordre d'implémentation des fonctionnalités). Les tests peuvent être réalisés soit par une personne interne au projet (développeur, CDP, expert), soit par une personne externe (camarade, famille, profs, ect…), soit via un test unitaire. Le personne/outils ayant réalisé le test sera mentionné dans le tableau de résultat.

5 RÉALISATION

5.1 Dossier de Réalisation

//Expliquer mise en place de git avec visual studio, mise en place du projet, framework, etc…
Qu'est qui a finalement été fait en comparaison avec la conception
Mise en place du git -> projet VS2022 (framework)->Classes->Code important ->DB
Se baser sur le git P_Appro2 pour ordre et éléments à voir

Utilisation de thread finalement (pour les inputs utilisateur, limitation du nombre d'input par frame, Management visuel pour les tetriminos)
Musique dans le menu (justification de l'utilisation de .Framework (using System.Media)).
Héritage sur les pièces (avec une méthode en virtual/override)
Utilisation de classe Static (pourquoi pas singleton ? => perte de temps sur le projet pour des détails mineurs qui apportent peu. Aucuns problèmes liés à une multiple instance de certaines classe). Citer les classes non-static et pourquoi elles ne sont pas static

Parler des classes
Gestion de la suppression des lignes
Parler un peu des problèmes survenus au cours du projet (et comment ils ont été résolus ou bien quel autre chemin a été pris)

Image pour montre le pseudo code parfois fait en amont pour aider à la réflexion

```
0 références
private bool CheckCanMoveInPlayZone()
{
    bool spaceIsEmpty = false;

    //TO DO: Check if next movement will result in the current form (1) to collided with an other form (2 or 3)
    //if no, space is empty and the piece can move
    //else, the piece can move in the wanted direction
    //Check the same for the rotation (Move the piece in the array BUT NOT visually, check, return result?)


    return spaceIsEmpty;
}
```

Figure 2 : Exemple de pseudo code fait avant de commencer certaines fonctionnalités

- Cette partie permet de reproduire ou reprendre le projet par un tiers.
- Pour chaque étape, il faut décrire sa mise en œuvre. Typiquement :
  - Versions des outils logiciels utilisés (OS, applications, pilotes, librairies, etc.)
  - Configurations spéciales des outils (Equipements, PC, machines, outillage, etc.)
  - Code source commenté des éléments logiciels développés.
  - Modèle physique d'une base de données.
  - Arborescences des documents produits.
- Il faut décrire le parcours de réalisation et justifier les choix.

5.2 Points supplémentaires

Cette partie présentera les ajouts supplémentaires fait au jeu ainsi qu'une éventuelle description détaillée de la manière utilisée

Ajout supplémentaire	Date	Description
Sons	02.05.2024	Ajout de musique dans le menu. Gestion de la musique (on/off) dans les options
Prévisualisation du prochain tetriminos	15.05.2024	Désormais, on peut voir quel tetriminos viendra une fois celui en jeu posé.

5.2.1 Prévisualisation du prochain tetriminos

Afin d'aider le joueur, on peut lui montrer quel tetriminos lui sera donné une fois le tetriminos actuel posé.
Pour cela, dans le TetrminosManager, une deuxième variable de type Tetriminos a été créée (nextTetriminos). Lorsqu'un new tetriminos est demandé, le tetrminos

Figure 3 : Zone de prévisualisation

actuel devient celui prévu au préalable et nextTetriminos est redéfini via un random. Pour éviter tout problème au lancement du jeu (dû au fait qu'il n'y pas de tetriminos défini pour la prévisualisation au début) le programme vérifie que nextTetriminos n'est pas null. Concernant l'affichage du tetriminos (visuellement), une zone dédiée a été ajoutée à côté de la zone de jeu.

- Historique des modifications demandées (ou nécessaires) aux spécifications détaillées.
- Date, raison, description, etc.

6 TESTS

6.1 Dossier des tests

6.1.1 Procédure

La procédure des tests est expliqué afin de comprendre la bonne démarche à effectuer.

6.1.1.1 Menu

Procédure de test pour le menu :

1. Tester des bouger haut en bas avec les flèches
2. Si la flèche est tout en bas ou tout en haut, elle revient au prochain choix possible (si tout en haut, revient tout en bas et inversement)
3. La touche enter permet d'accéder au sous-menu
4. Le sous-menu est celui attendu
5. Il est possible de revenir en arrière

6.1.1.2 Options

Procédure de test pour les options:

1. La touche enter permet de changer l'option sélectionnée
2. L'option passe correctement au prochain choix (ON/OFF pour le son (voir couleur), autre possibilité pour les autres)
3. Les options choisies sont les bonnes une fois en jeu. (La musique s'active directement, les touches sont les bonnes. Pour la difficulté, voir vitesse du jeu ou questions (voir DB))

6.1.1.3 Pseudo

Procédure de test pour le pseudo :

1. Un pseudo aléatoire est attribué lors du premier lancement.
2. Il est unique (voir en DB)
3. Il reste le même tant qu'il n'est pas modifié dans le param.txt

6.1.1.4 Déplacement des pièces

Procédure de test pour le déplacement des pièces :

1. Les pièces peuvent se déplacer grâce aux touches choisies dans les options
2. Elles ne sortent pas de la zone de jeu
3. Elles descendent naturellement toutes seules
4. Elles s'empilent correctement
5. Elles peuvent tourner sur elles-mêmes

6.1.1.5 Questions

Procédure de test pour les questions:

1. Lors de la complétion d'une ligne, une question est posée
2. Elle provient de la DB (voir DB)
3. Elle est du bon niveau de difficulté (voir DB)
4. Le joueur peut y répondre
5. En cas de bonne réponse, elle devient verte et des points sont attribués
6. En cas de mauvaise réponse, elle devient rouge, la bonne réponse s'affiche et une ligne se bloque
7. Une marge d'erreur est accordée au joueur selon la longueur de la réponse attendue (min 12 charactères)

6.1.1.6 Score

Procédure de test pour le score :

1.

6.1.1.7 Game Over

Procédure de test pour le Game Over :

1.

6.1.1.8 DB

Procédure de test pour la DB:

1.

6.1.2 Résumé des tests

Cette partie résumera les tests effectués et permettra d'effectuer un bilan de ceux-ci.

Nom du test	Date du test	Passation	Commentaire
Menu	13.05.2024	OK	Il est possible de parcourir tout le menu et de de revenir en arrière. Le sous-menu « score » est plus lent si la DB n'est pas accessible mais reste fonctionnel.
Options	13.05.2024	OK	Il est possible de modifier chacune des options. Celle-ci est directement enregistrée dans le fichier de paramètre et prend effet immédiatement (pour le son par exemple).
Pseudo	17.05.2024	OK	(Peut être amélioré) Un pseudo aléatoire est attribué au joueur. Le joueur ne peut pas encore le modifié.
Déplacement des pièces	17.05.2024	OK	Les pièces peuvent être déplacées

			horizontalement et vers le bas (touche du bas ou espace pour faire descendre directement). Elles ne sortent pas de la zone de jeu. Elles peuvent également tourner (il reste un bug si la pièce est tout en bas, elle sort de la zone si on la tourne, le jeu ne crash pas)
Questions	17.05.2024	OK	Les questions sont prises depuis la BD. Elles sont ensuite choisies en fonction de leur niveau de difficulté. Une question aléatoire est posée au joueur.
Score	17.05.2024	OK	Le score augmente si le joueur répond correctement à une question. Plus la question est dure, plus le joueur gagne de points
Game over	17.05.2024	OK	Si une pièce dépasse la ligne rouge, l'écran de fin est affiché avec le score final, le nombre de bonnes et de mauvaises réponses et la possibilité de retourner au menu principal
DB	17.05.2024	OK	Les informations sont bien récupérées depuis la DB. Les parties sont enregistrées en DB. En cas d'erreur avec celle-ci, le jeu ne crash pas et l'erreur est notée dans un fichier de log.

- On dresse le bilan des tests effectués (qui, quand, avec quelles données…) sous forme de procédure. Lorsque cela est possible, fournir un tableau des tests effectués avec les résultats obtenus et les actions à entreprendre en conséquence (et une estimation de leur durée).
- Si des tests prévus dans la stratégie n'ont pas pu être effectués :
  - raison, décisions, etc.
- Liste des bugs répertoriés avec la date de découverte et leur état:
  - Corrigé, date de correction, corrigé par, etc.

7 CONCLUSION

7.1 Bilan des fonctionnalités demandées

//tableau bilan final (qu'est qui a été fait, qu'est-ce qui reste à faire, pourcentage et temps restant)

- Il s'agit de reprendre point par point les fonctionnalités décrites dans les spécifications de départ et de définir si elles sont atteintes ou pas, et pourquoi.
- Si ce n'est pas le cas, estimer en « % » ou en « temps supplémentaire » le travail qu'il reste à accomplir pour terminer le tout.

7.2 Bilan de la planification

//comparaison entre planif et réalité, expliquer les deltas importants

- Distinguer et expliquer les tâches qui ont généré des retards ou de l'avance dans la gestion du projet. Indiquer les différence entre les planifications initiales et détaillées avec le journal de travail.

7.3 Bilan personnel

- Si c'était à refaire:
  - Qu'est-ce qu'il faudrait garder ? Les plus et les moins ?
  - Qu'est-ce qu'il faudrait gérer, réaliser ou traiter différemment ?
- Qu'est que ce projet m'a appris ?
- Suite à donner, améliorations souhaitables, …
- Remerciements, signature, etc.

8 DIVERS

8.1 Table des illustrations

8.2 Journal de travail

- Date, activité (description qui permet de reproduire le cheminement du projet), durée, liens et références sur des documents externes. Lorsqu'une activité de recherches a été entreprise, il convient d'énumérer ce qui a été trouvé, avec les références.

8.3 Webographie

//tableau avec nom du site et lien de la page concernée

- Références des sites Internet consultés durant le projet.

9 ANNEXES

//git hub (public), installeur ?, JRNLTRV, DB, doc utilisateur (mode d'emploi) ?,

- Listing du code source (partiel ou, plus rarement complet)
- Guide(s) d'utilisation et/ou guide de l'administrateur
- Etat ou « dump » de la configuration des équipements (routeur, switch, robot, etc.).
- Extraits de catalogue, documentation de fabricant, etc.