EPS Escola Politècnica Superior
UdG

Projecte/Treball Fi de Carrera

Estudi: Eng. Tècn. Informàtica de Sistemes. Pla 2001

Títol: Generació semi-automàtica de façanes procedurals

Document: Memòria

Alumne: Rachid Mahmoudi

Director/Tutor: Gustavo Patow
Gonzalo Besuivsky
Departament: Informàtica i Matemàtica Aplicada
Àrea: LSI

Convocatòria (mes/any): Setembre 2012

Agraïments

En primer lloc voldria agrair a Gustavo Patow i Gonzalo Besuievsky per l'oportunitat que m'han ofert per realitzar aquest projecte i aprendre d'ell, i al Departament d'Informàtica i Matemàtica Aplicada a permetre realitzar-lo.

Als meus pares, perquè quan em poso capgròs ells saben orientar, i els seus consells sempre m'han ajudat. I per descomptat a la resta de la meva família, les meves germanes, els meus oncles, cosins, ... per la vostra confiança en mi.

A tots els meus professors, des del col·legi fins a la universitat, i en especial als meus tutors en Gustavo i en Gonzalo i tots els professors de la universitat de Girona, per tot el que he après gràcies a vosaltres.

A tots els meus companys i amics de la universitat, sobretot a Llorenç Burgas i Borja Giner Triguero, perquè sense tots vosaltres, els vostres resums, vostres consells, i la vostra ajuda segur que no estava escrivint aquestes línies.

Gràcies.

Índex de continguts

Capítol 1: Introducció

El modelatge procedural i visualització de sistemes complexos com les ciutats és una gran repte per als gràfics per ordinador. Els edificis són sistemes d'alta complexitat funcional i visual. Actualment existeix un gran interès en l'automatització en el modelatge de edificis urbans (actuals, històriques o virtuals), degut a més demanda en indústries com la del cinema, la realitat virtual i els videojocs. i a la xifra del capital que generen en el àmbit de l'oci.

La part d'aquestes estructures arquitectòniques exteriors que més detalls conté i que, per tant requereix més atenció, són les façanes, A més, és molt difícil definir les regles necessàries per generar façanes a partir d'una imatge, ja que és un problema obert pel que no existeix solucions estàndards ni de codi lliure.

Un dels enfocaments més usuals avui dia, dels que eviten la feina manual, és l'ús de tècniques procedurals per a la descripció d'un edifici o estructura arquitectònica.

El modelatge procedural pretén solucionar gran part d'aquest problema utilitzant les funcionalitats que té per tal de generar de forma automàtica les façanes que es necessitin. El procés consisteix en deixar que l'ordinador faci la feina complicada, d'aquesta forma l'usuari simplement ha d'especificar els diferents paràmetres per tal d'aconseguir el resultat desitjat.

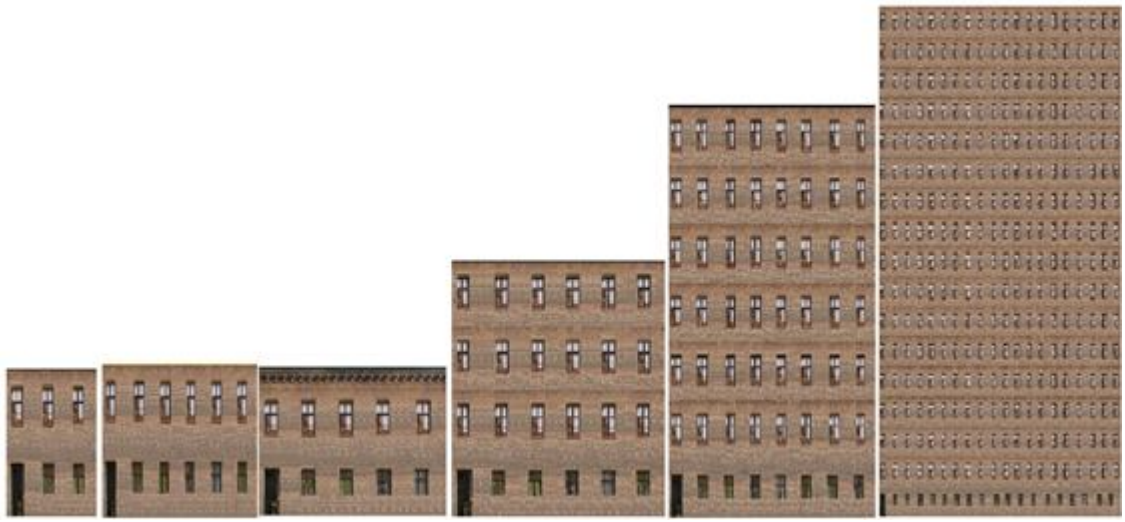


Figura 1: generació de façanes

1.1 Motivacions Personals

Una de les assignatures que em va agradar molt es visió per computador. Ja des del principi vaig buscar un projecte relacionat amb aquesta temàtica, perquè volia aprofundir més, ja que només vaig treballar a una assignatura, i amb un projecte d'aquest tipus podia aprofundir més en aquesta branca de la informàtica.

1.2 Propòsits i objectius

L'objectiu d'aquest PFC és desenvolupar una eina d'edició de façanes procedural a partir d'una imatge d'una façana real. L'aplicació generarà les regles procedurals de la façana a partir de dades adquirides del model que es vol representar, com una fotografia. L'usuari de l'aplicació generarà de forma semi-automàtica i interactiva les regles de subdivisió i repetició, especificant també la inserció de elements arquitectònics (portes, finestres), que podran ser instanciats a partir d'una llibreria. Un cop generades, les regles s'escriuran en el format del sistema BuildingEngine per integrar-se completament dins del procés de modelatge urbà.

Aquest projecte es desenvoluparà en Matlab. Podem organitzar la feina seguint una estructura modular, el que implica:

- Aprendre i manejar el llenguatge de programació Matlab
- Desenvolupament d'un sistema de detecció de finestres, portes, ... en 2D.
- Definició i Estudi d'un conjunt inicial d'estructures de base que es combinaran per crear la forma bàsica d'una façana.
- Generació les regles procedurals de la façana a partir de dades adquirides del model que es vol representar, com una fotografia en 2D.
- Aplicació d'aquest conjunt de regles sobre una forma bàsica que generi iterativament les diferents parts de la façana: portes, finestres, etc
- Test d'aplicació amb una façana senzilla amb repeticions molt regulars.
- Aplicació a façanes més complicades amb diferents finestres i patrons de repeticions més complicats.
- Desenvolupament d'una sèrie de llibreries per tal de instanciar interactivament els elements de façana més comuns d'edificis.
- Finalització i arrodoniment de la documentació del treball realitzat.

Estudi d'alternatives:

- Aprendre i manejar el llenguatge de programació Python.
- Estudi de les llibreries wxPython

1.3 Estructuració de la documentació

Aquesta memòria està estructurada en catorze capítols fonamentals que recullen tota la documentació i explicacions necessàries per la completa comprensió del projecte.

1. Introducció, motivació, propòsits i objectius del projecte. En aquest capítol s'explicaran el perquè del desenvolupament d'aquest projecte, quins són els objectius proposats i com s'ha organitzat el desenvolupament.

2. Estudi de la viabilitat. En aquest capítol es justifiquen els paràmetres que fan possible el desenvolupament del projecte.

3. Metodologia. Aquest capítol conte l'explicació de la metodologia utilitzada.

4. Planificació. En aquesta etapa es defineix l'estratègia seguida per arribar als objectius plantejats.

5. Marc de treball i conceptes previs. En aquest capítol es descriuen els diversos aspectes relacionats amb el desenvolupament general del projecte, que ajudaran a entendre millor els següents capítols. També es tractaran les principals accions desenvolupades durant les primeres etapes de la realització del projecte. S'inclouran els passos d'estudi i aprenentatge de conceptes que s'hagin utilitzat per desenvolupament.

6. Requisits del sistema. En aquest capítol es defineixen els requeriments del software, les quals recullen, a grans trets, els objectius de l'aplicació juntament amb les funcionalitats que es volen obtenir. Aquest document permet entendre els elements que rodegen al sistema informàtic que s'intenta construir.

7. Estudis i decisions. Aquesta secció conté una descripció de les eines utilitzades, amb les seves característiques i l'ús que se'ls hi ha donat.

8. Anàlisi i disseny del sistema. Aquest apartat proporciona una comprensió precisa de les necessitats del sistema, es a dir, s'encarrega

de la investigació del problema a resoldre, però no s'interessa a trobar una solució. Fent servir l'enginyeria del software, aquesta secció es tradueixen els requeriment nombrats en capítols anteriors a un llenguatge més formal. La part del disseny permet augmentar el nivell d'especificació, y realitzar un esquema d'implementació del sistema mitjançant diverses eines de programació orientada a objectes, explicant les classes i els mètodes que formen el projecte.

9. Implementació i proves. En aquest capítol es donen a conèixer com s'ha implementat l'aplicació les classes i els mètodes que resulten més significatius per a la comprensió del funcionament de l'aplicació.

10. Implementació i resultats. En aquest capítol es mostren proves d'execució de l'aplicació, es mostren façanes generades amb diferents paràmetres a partir de la façana d'entrada.

11. Conclusions. En aquest apartat s'exposaran les conclusions obtingudes un cop finalitzat el projecte.

12. Treball futur. En aquest secció s'exposa tot allò que es podria millorar en l'aplicació, o ampliar-la de forma interessant.

13. Bibliografia. Aquest capítol conté les referències utilitzades per al desenvolupament del projecte.

14. Annexos. Aquesta secció conté les definicions dels tecnicismes més freqüentment utilitzats, així com explicacions i experiments no indispensables per a la comprensió global del projecte.

15. Manual d'usuari i instal·lació. Aquesta secció inclou les especificacions del funcionament del producte.

Capítol 2: Estudi de la viabilitat

Per desenvolupar el projecte no es requereix d'una gran infraestructura i els costos d'estructura són limitats.

El llenguatge de programació emprat ha estat el llenguatge M (Matlab).

Aquest PFC s'ha desenvolupat sota l'entorn Windows, utilitzant eines de desenvolupament multiplataforma. A més aquestes eines estan disponible en plataformes Unix, Windows i Mac OS X.

2.1 Recursos Humans

En aquest projecte només ha estat necessari la participació d'un desenvolupador de software capaç de projectar les idees inicials amb codi. També es necessari un Analista de programari. Tots dos rols, tant el de programador com el d'analista de programari, els he assumit jo.

2.2 Avaluació prèvia de costos i medis

Són tres estudis de viabilitat necessaris pel desenvolupament del sistema: la viabilitat econòmica, la viabilitat tècnica i la viabilitat legal. Degut que aquesta aplicació no presenta grans riscos i cap problema legal, només considerarem les següents àrees:

- Estudi de viabilitat tecnològica.
- Estudi de viabilitat econòmica.

2.2.1 Estudi de viabilitat tecnològica

L'estudi de la viabilitat tecnològica comença amb una definició tècnica del sistema proposat, donant resposta a les següents preguntes: Quina tecnologia es necessitarem per aconseguir la funcionalitat de l'aplicació? Quins nous materials, mètodes o processos es requereixen? Com afectaran al cost aquests elements de tecnologia?

Per sort, ja es disposen dels dispositius hardware necessaris per poder realitzar el projecte, és a dir, un ordinador capaç de realitzar tota la programació, la

interpretació del codi i la visualització dels resultats, i la llicencia del software Matlab.

2.2.2 Estudi de viabilitat econòmica

La valoració de l'anàlisi econòmica serà separada en dues parts: els costos de recursos humans i els costos de la maquinària.

2.3 Costos de recursos humans

Per calcular els costs dels recursos humans s'ha associat un perfil o rol de treballador per cadascun de les tasques.

Cost analista: 20 € / hora

Cost programador: 15 € / hora

Tasca	Perfil	Hores	Cost
Llegir articles	Analista	30	20
Estudi de Python	Analista	15	300
Estudi de wxPython	Analista	15	300
Disseny d'algorismes	Analista	50	1000
Implementació codi	Programador	150	2250
Estudi de Matlab	Analista	10	200
Implementació codi	Programador	300	4500
Implementació interfície	Programador	50	750
Proves i optimització	Programador	20	300
Memòria	Analista	100	2000
Total		**740**	**12200**

No he tingut cap despesa per hardware, ja què des d'un principi ja es tenia tots els components necessaris, i en el transcurs del projecte no s'ha necessitat cap hardware nou. Tot i així, s'ha comptabilitzat l'amortització del hardware:

$$Amortització = \frac{Cost\ hardware}{Temps\ d'amortització} . Temps\ de\ projecte$$

He definit que el temps d'amortització són 3 anys, el temps del projecte 1 any i el cost del hardware, o sigui, el meu portàtil, és de 499€. Això dóna que he tingut una despesa d'amortització de 166€.

Al no haver de pagar llicència per programari, però si per software que es 40.96€, i tenint en compte l'amortització, el cost total del projecte, incloent-hi els costos de programació, ha estat de 12407€

Capítol 3: Metodologia

3.1 Metodologia skylineEngine

Per a la realització d'aquest projecte no s'ha seguit una metodologia de treball estàndard, sinó que es s'ha triat i utilitzat una metodologia personalitzada i que va ser plantejada amb el tutor. Aquesta metodologia és la mateixa que se segueix en el projecte **skylineEngine**, i els passos que segueix són els següents:

0 - Escollir el treball a desenvolupar.

1 - Decidir el llenguatge de programació i eines a utilitzar.

2 - Aprendre el llenguatge de programació i les eines escollides.

3 - Estructurar el treball en parts segons les funcions que s'hagin de realitzar.

4 - Desenvolupar la part corresponent seguint l'ordre de l'estructura del treball.

5 - Fer comprovacions per confirmar que el funcionament és correcte en finalitzar cada part.

- Si en fer les comprovacions el resultat no és el desitjat, es tornarà al punt 4 per realitzar els canvis oportuns en l'última part desenvolupada o en les anteriors, si és necessari.
- Si en fer les comprovacions el resultat és el desitjat, es desenvoluparà la següent part tornant al punt 4. Una vegada s'hagin finalitzat totes les parts amb els seus respectives comprovacions, s'iniciarà el punt 6.

6 - Unir totes les parts desenvolupades i comprovar que el funcionament és correcte.

- Si en fer les comprovacions el resultat no és el desitjat, es tornarà al punt 4 per realitzar els canvis oportuns en l'última part desenvolupada o en les anteriors, si és necessari.
- Si en realitzar les comprovacions el resultat és l'esperat, s'iniciarà el punt

7 - Generar diferents models d'exemple per comprovar que el funcionament és el correcte.

- Si en fer les comprovacions el resultat no és el desitjat, es tornarà al punt 4 per realitzar els canvis oportuns en l'última part desenvolupada o en les anteriors, si és convenient.
- Si en realitzar les comprovacions el resultat és l'esperat, s'iniciarà el punt

8 - Presentar documentació.

Tal i com es pot veure, consisteix en dividir el projecte en mòduls i organitzar en el temps el desenvolupament, el temps de verificació i de correcció. Tant

durant el temps de desenvolupament com de verificació es fa un seguiment mitjançant tutories setmanals o bisetmanals depenent de l'etapa, ja que en els inicis la manera d'avançar és molt més lenta que al final i no sempre es necessari fer tutories setmanalment. Quan s'acaba un mòdul, sempre que es pugui, es finalitza totalment de manera que no es torna a tocar, així garantim que les errors que puguin sorgir en el mòdul actual són únicament d'aquest i no de cap dels anteriors, o almenys que afectin el mínim possible.

La tècnica que s'ha seguit per a la creació d'aquesta aplicació és la de disseny descendent. Per això hem utilitzat com a metodologia el llenguatge UML (Llenguatge de Modelatge Unificat o Unified Modeling Language), que tot i no ser una metodologia, és un llenguatge de modelat estàndard pel camp de l'enginyeria del programari. L'UML s'utilitza per definir un sistema, per detallar els seus elements, per documentar i construir. Per aconseguir això, l'UML disposa de nombrosos tipus de diagrames que mostren diversos aspectes dels elements representats.

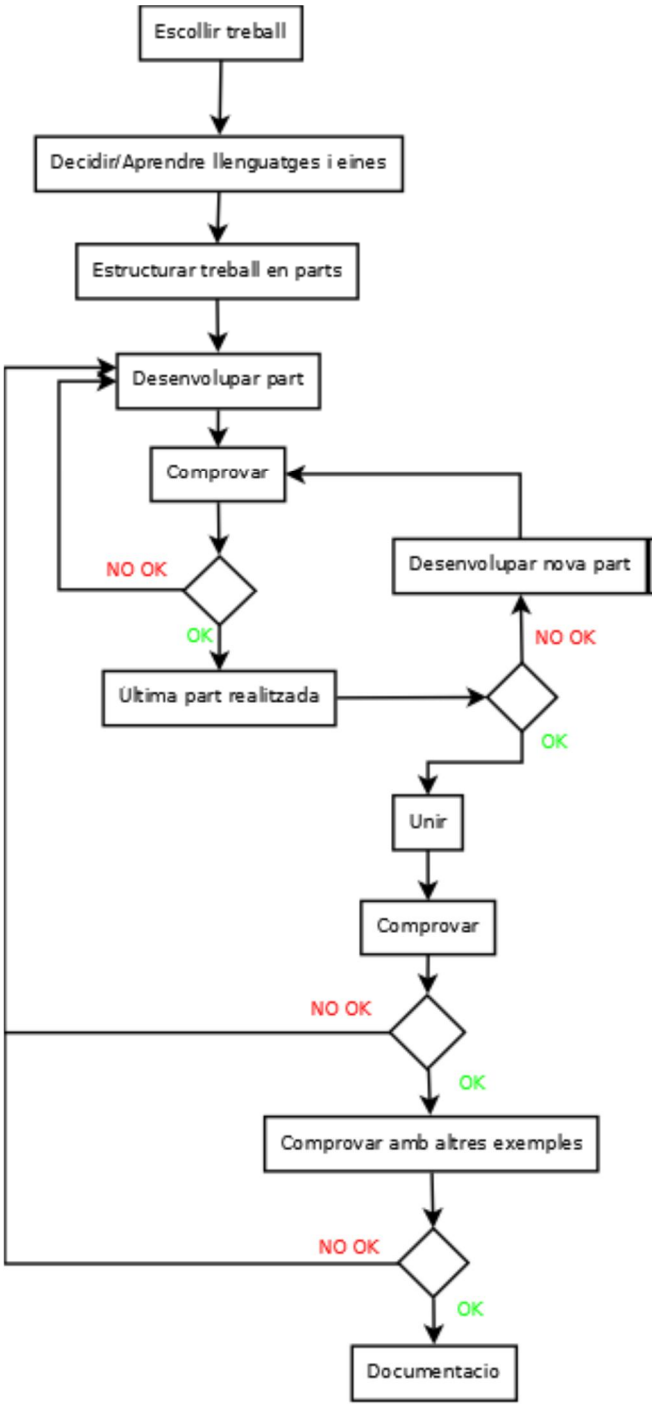


*Figura 2: Diagrama de flux de la metodologia **skyline**Engine*

Capítol 4: Planificació

Aquest projecte es va iniciar a mitjans de desembre del 2011 i s'ha acabat al final de agost del 2012. Inicialment s'havia programat per acabar-lo al juliol del 2012. Però, per motius explicats en l'apartat 7.1, no s'ha pogut fer-lo.

La primera etapa, fase d'aprenentatge d'eines que utilitzarem, correspon a l'etapa inicial de llegir els articles el que van triar els tutors i els que s'ha trobat per internet, i intentar entendre o al menys sacar uns punts d'interès, ja que no era fàcil entendre un article a llegir lo una vegada. La següent etapa era presa de contacte amb les eines necessàries per poder desenvolupar el projecte. Un cop aclarits els conceptes bàsics, el següent pas ha estat introduir-se en mon de Matlab, lectura de manuals, lectura de codi i execució d'exemples, etc.

Un cop assimilats tots aquests conceptes, el següent pas consisteix en començar a implementar els algoritmes corresponents a la primera tasca. L'objectiu d'aquesta fase consisteix en detectar finestres i portes en la façana.

La segona fase del projecte, després de tenir la façana definida en tots els seus components (finestres, portes, etc), és genera un fitxer .txt corresponent a la imatge, descrivint tots els trossos amb el nom (amb format .jpg) i la mida.

L'objectiu de la tercera tasca consisteix en generar façanes amb paràmetres diferents a partir de la primera. Els paràmetres de entrada són el número de pisos i el número de finestres que es vol construir.

Amb el resultat de la tercera fase, ja es pot fer la quarta fase: generar un fitxer .txt corresponent a la imatge generada, descrivint tots els trossos amb el nom (amb format .jpg) i la mida de cadascun, La següent fase és fer les interfícies d'usuari i la correcció d'errors.

Per últim, cal la verificació i proves finals dels algoritmes implementats. En aquesta fase es fan comprovacions que els algoritmes facin realment el que han de fer en situacions més generals, i corregir els possibles errors que hi hagin.

Finalment, els últims mesos han estat dedicats a corregir errors que hagin anat sorgint, millora de codi i a arrodonir i organitzar la memòria d'aquest projecte.

4.1 Temps estimat

Inicialment es va planejar que el projecte dura 7 mesos i la distribució que es va elegir en un primer moment va ser la que es pot observar en la figura 3:

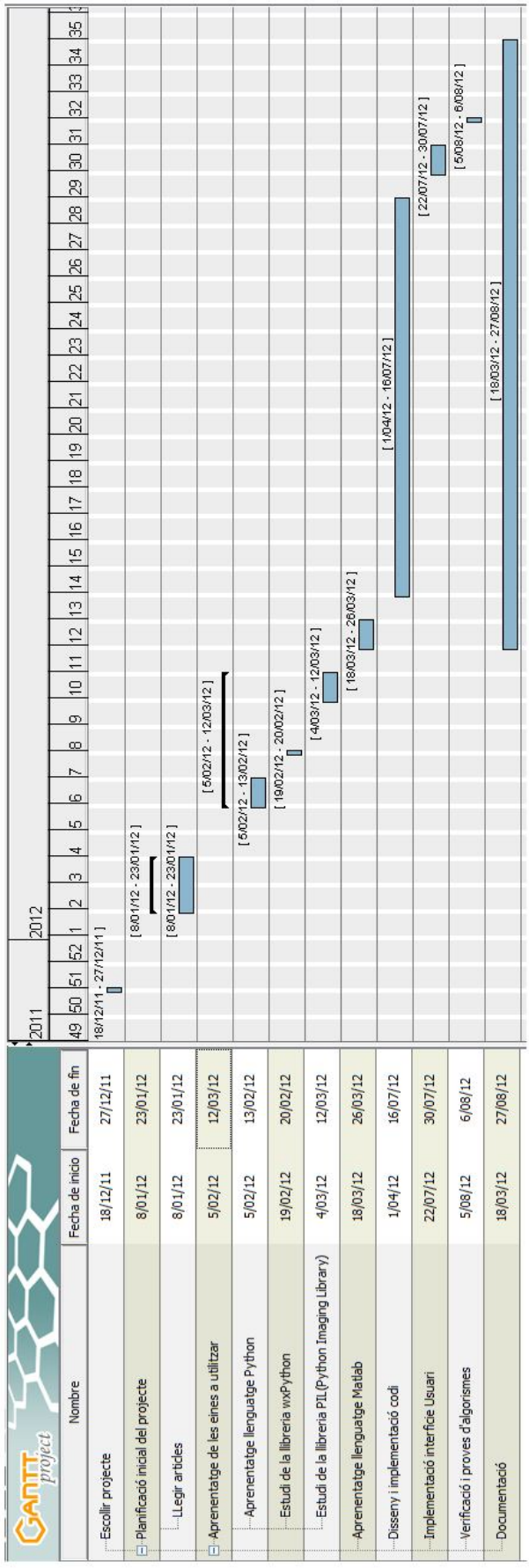


Figura 3: Diagrama de Gantt de la planificació inicial

4.2 Resultat estimats

S'espera crear una aplicació totalment funcional, que pugui ser reutilitzada en la seva totalitat, complint totes les expectatives i objectius plantejats en els apartats anteriors. Així, l'usuari podrà generar de forma automàtica diferents façanes, entrant només els paràmetres que necessiti el programa.

Capítol 5: Marc de treball i conceptes previs

En aquest apartat seran detallats els conceptes necessaris per tal de poder entendre els algorismes desenvolupats i les eines utilitzades per poder dur a terme amb èxit aquest projecte.

5.1 Modelatge procedural

Modelatge procedural es refereix a una àmplia varietat de tècniques per a la creació de models automàtics, pel que es pot utilitzar per crear models que siguin molt complexes. Els exemples més comuns inclouen els objectes naturals com arbres, el paisatge (muntanyes ...), els núvols, etc. També és possible utilitzar el modelatge procedural per a objectes artificials com ara edificis, i fins i tot ciutats.

El modelatge procedural es defineixen per un petit conjunt de dades que descriu les propietats generals del model. Per exemple, un arbre pot ser definit per algunes propietats de ramificació i formes de fulles El model real es construeix mitjançant un procediment que utilitza, sovint l'aleatorietat per afegir varietat. D'aquesta manera, un patró d'arbre es pot utilitzar per modelar un bosc sencer.

Figura 4: un edifici creat amb Modelatge Procedural

5.2 SkylineEngine

skylineEngine és un sistema de modelatge procedural urbà, desenvolupat pel Grup de Geometria i Gràfics de la Universitat de Girona. És una eina en evolució continua, que integra diferents mòduls.

skylineEngine és una eina de modelatge procedural urbà desenvolupat com un banc de proves per a nous algoritmes i tècniques de modelatge urbà. Aquest sistema presenta algunes característiques noves que el converteixen en un sistema únic, com un paradigma basat en grafs que permet a l'usuari crear tant contingut ric en ciutats amb districtes diferents, carreteres principals i secundàries, els blocs, lots i edificis, com també altres elements urbans com carrers, voreres, parcs, ponts i monuments.

Tot en **skyline**Engine està fet per mòduls, alguns d'ells en codi Python, altres en forma de Houdini Digital Assets (HDA's). Dintre del **skyline**Engine, el mòdul buildingEngine ha estat concebut com a una eina de modelatge procedural d'edificis i està implementat Houdini i codi escrit en Python.

Figura 5: La ciutat de Girona generada amb **skyline**Engine

Capítol 6: Requisits del sistema

El projecte en sí, ha estat pensat perquè depengui al mínim d'uns requisits específics. Tot i això, hi han uns requeriments mínims per tal de poder seguir desenvolupant el projecte, o per simplement usar l'aplicació resultant.

Requisits des del punt de vista d'un usuari final:

- Disposar d'un ordinador amb:
  - Sistema operatiu a escollir segons les preferències de l'usuari.
  - Capacitat suficient per emmagatzemar les imatges generades durant 1 sessió.
  - Tenir Matlab instal·lat en la maquina.

Requisits des del punt de vista d'un desenvolupador:

- Disposar d'un ordinador amb:
  - Sistema operatiu a escollir segons les preferències de l'usuari.
  - Capacitat suficient per emmagatzemar i compilar totes les llibreries necessàries. Pot arribar a necessitar-se segons la modalitat de complicació i els exemples que es desitgin fins a un 1Gb. Les llibreries bàsiques per poder seguir desenvolupant el projecte estan incloses en Matlab.
  - Una tarja gràfica actual. No és necessari disposar d'una tarja gràfica d'alt rendiment, tot i que si es disposa d'una bona milloraran considerablement els resultats.

6.1 Requeriments funcionals

En aquest apartat es descriuran els serveis que oferirà aquesta aplicació, sense tenir en compte la seva implementació.

Els requisits funcionals són els que indiquen què ha de fer l'aplicació. S'han definit els següents:

- Mostrar els resultats dels nostres algorismes en Matlab.
- Modificar en temps real les característiques dels diferents nodes des de la interfície d'usuari.

S'entenen com a requeriments generals del sistema les principals funcionalitats del software a desenvolupar, reflectant en tot moment les responsabilitats del programa construït.

A continuació hi ha els principals requeriments:

Totes les sortides del programa que se mostren són a partir d'aquesta imatge d'entrada:

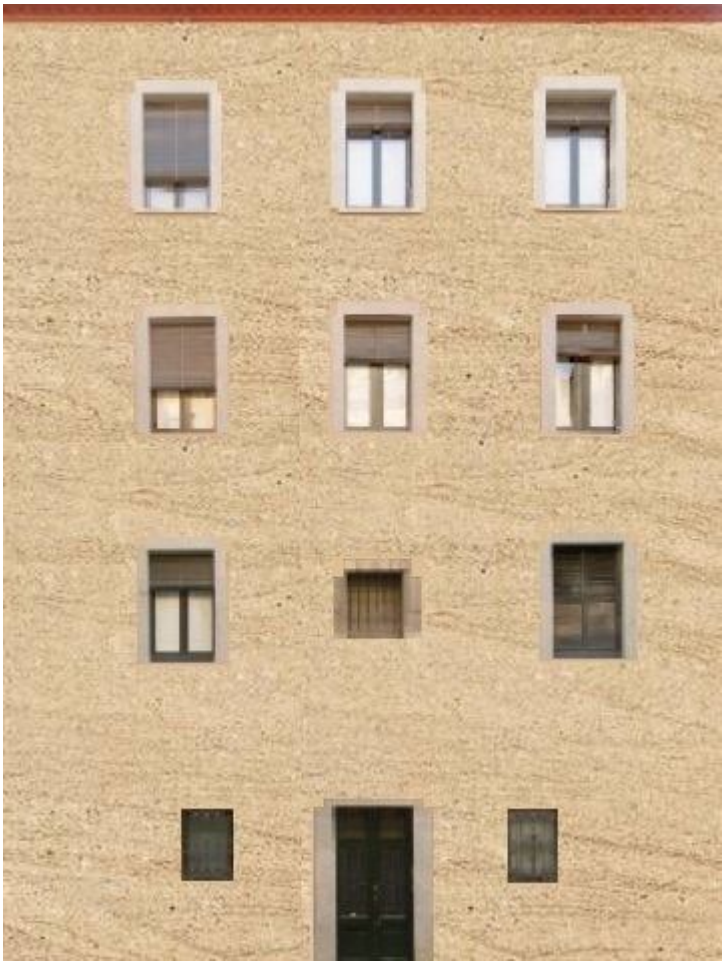


Figura 6: Imatge d'una façana real

- Detectar totes les finestres i portes que hi ha en la façana

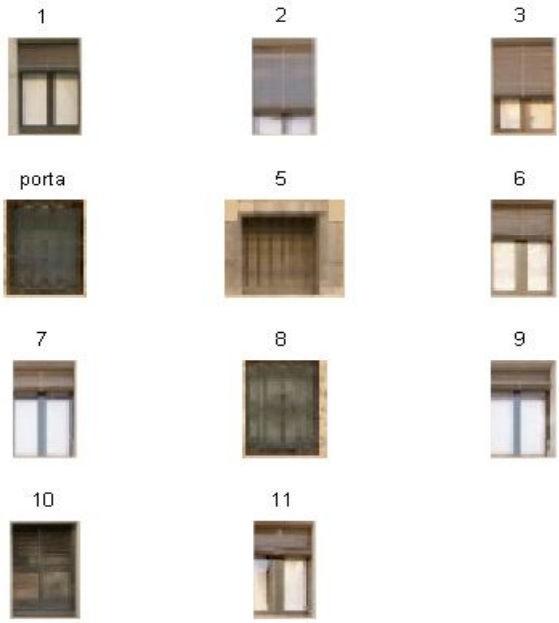


Figura 7: components trobades en la façana

- Dividir la façana en pisos

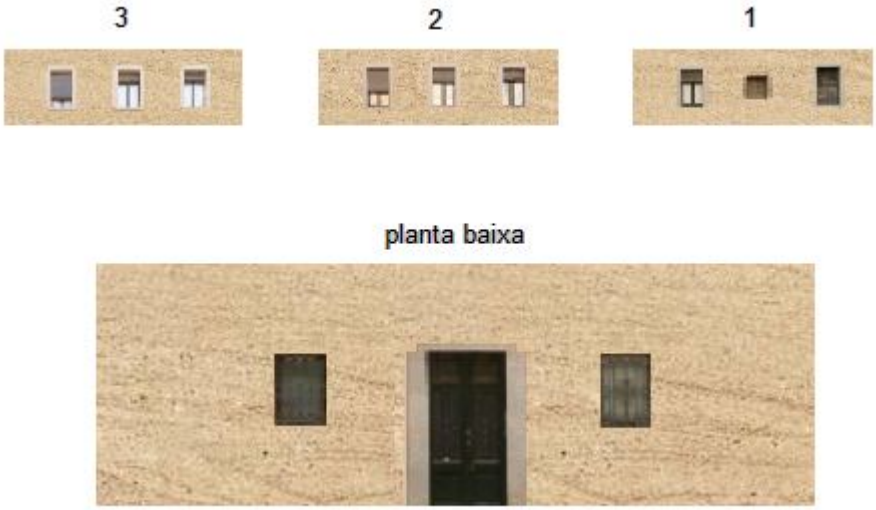


Figura 8: pisos detectats de la façana

- Dividir en columnes: columnes amb finestres i amb textura

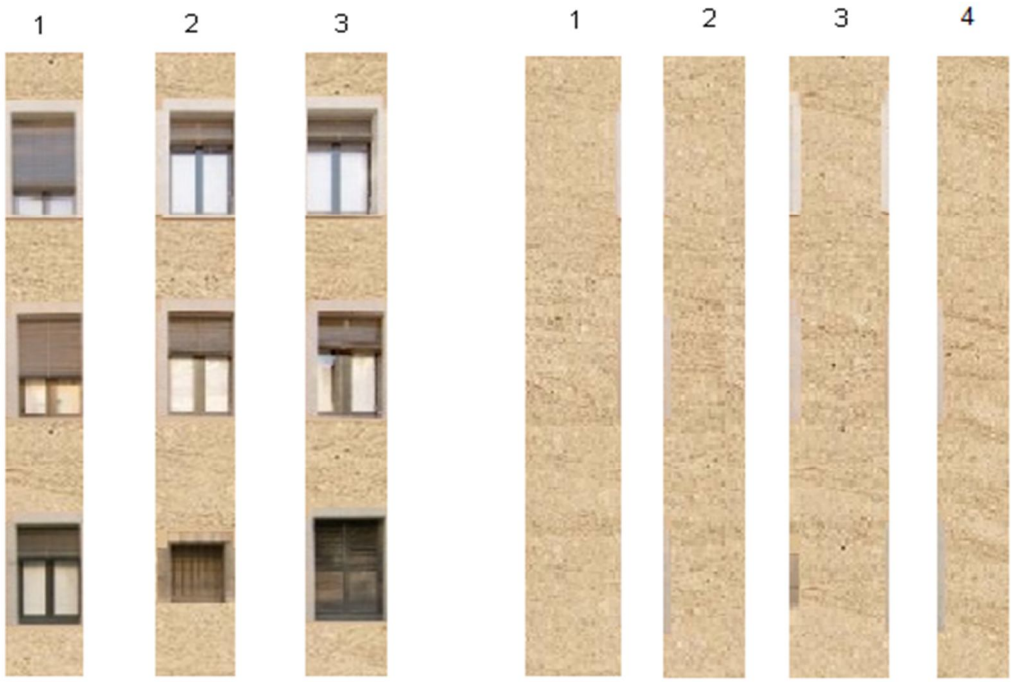


Figura 9: Columnes amb finestres i columnes de textura no inclòs la planta baixa

- Generar un fitxer .txt corresponent a la imatge, descrivint tots els trossos amb el nom (amb format .jpg) i la mida de cadascun *(l'annex 1 i 2).*

- Generar façanes a partir de paràmetres d'entrada

Figura 10: imatge generat a partir de paràmetres d'entrada (número de pisos igual a 10 i número de finestres per planta igual a 11)

- Guardar la imatge generada
- Generar un fitxer .txt de la imatge generada, descrivint tots els elements identificats amb el nom (amb format .jpg) i la mida de cadascun *(l'annex 3 i 4).*

6.1.1 Imatge regular

És una imatge amb alt grau de simetria, tal que es compleixen aquestes tres condicions (veure figura 11).

1. diferència brusca entres els píxels de la textura i els elements de la imatge
2. un espai entre els elements que estén en la mateixa columna
3. un espai entre els pisos

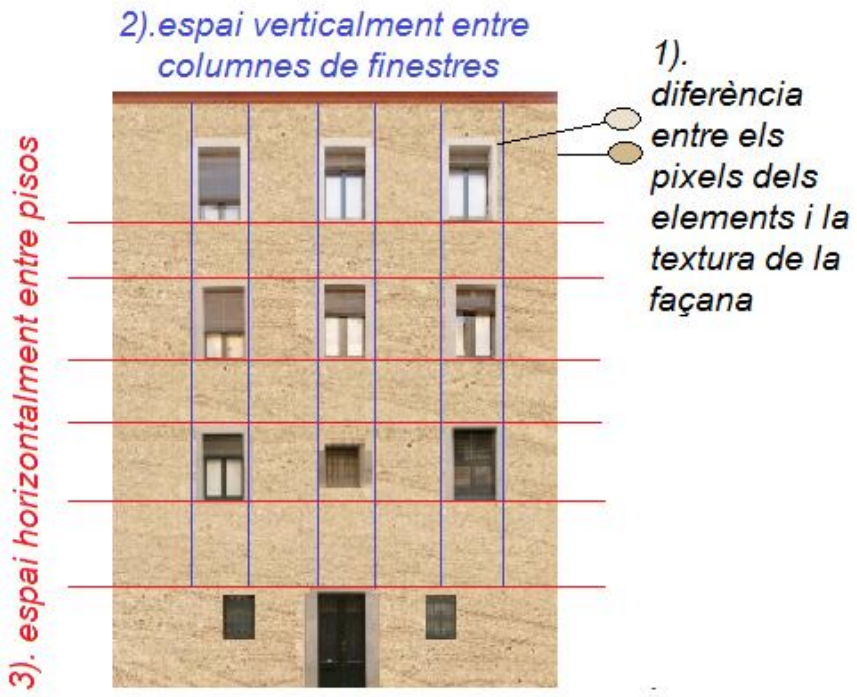


Figura 11: Un exemple d'una façana regular

- **Exemples de façanes regulars**

A continuació veiem un parell d'exemples de façanes regulars (figura 12).

Figura 12: façanes regulars

- **Exemples de façanes irregulars**

A continuació veiem un parell d'exemples de façanes irregulars (figura 13).

Figura 13: façanes irregulars

6.2 Requeriments no funcionals

Els requeriments no funcionals, en canvi, descriuen aspectes sobre com ha de ser el programa i no sobre què ha de fer. S'han definit els següents:

- El programa ha de ser eficient: no fer càlculs redundants ni utilitzar més memòria de la necessària.
- També hauria de ser extensible, permetent incorporar noves variants d'algorismes a mesura que es vagin desenvolupant.
- La interfície ha de ser intuïtiva i usable.
- S'ha de fer una documentació sobre el funcionament bàsic del programa.
- Hauria de poder funcionar en diversos sistemes operatius.

Com que la plataforma on es va desenvolupar es el Matlab, els requeriments mínims són els de Matlab, el qual és un software multi-plataforma ja que es pot executar per GNU/Linux, MS Windows o Mac OS X, i en el apartat de hardware 1GB de RAM com mínim, processador de 32 o 64 bits, 5 GB de disc dur, i un ratolí estàndard.

La implementació i les proves de l'aplicació s'ha fet amb un equip de les següents característiques:

- Processador Inter i3-2330M
- 4 GB de RAM
- Sistema operatiu Windows 7 64bits
- Targeta gràfica Intel HD

L'aplicació està programada amb llenguatge Matlab.

Capítol 7: Estudis i decisions

7.1 Justificació de l'entorn de desenvolupament

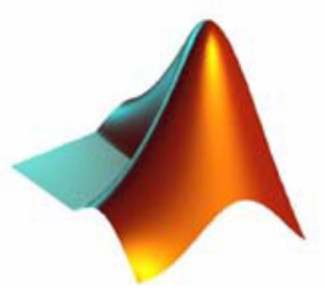
L'entorn de desenvolupament triat ha de complir diversos requisits i en funció d'aquests requisits es va decidir triar com a llenguatge de programació el llenguatge Matlab amb les llibreries corresponents.

Aquestes facilitats o requisits mínims que l'entorn de desenvolupament triat ha de proveir són:

a) Ser un entorn de desenvolupament d'última generació, provat, multiplataforma i amb documentació suficient sobre el funcionament.
b) Oferir un bon rendiment en les operacions de tractament digital d'imatges. Avui dia, les aplicacions de tractament digital d'imatges i totes les tècniques utilitzades per al processament, constitueixen dures proves al rendiment dels sistemes informàtics que les alberguen. Això es deu, principalment, a la complexitat dels algorismes utilitzats per a la seva implementació, així com a la quantitat d'informació que ha de mantenir-se en memòria, pensem que una petita imatge de mida 300 x 200, la qual utilitza un byte per a la representació de cada píxel ocupa, aproximadament, 60 Kbytes d'emmagatzematge. Pel que fa a la complexitat dels algorismes, pensem en el cost que suposa un bucle que faci un recorregut per aquesta imatge.
c) Posseir llibreries de rutines per a les operacions bàsiques de tractament digital d'imatges, la qual cosa permetria centrar el nostre treball en els objectius marcats, aconseguint millors resultats.
d) Ser obert, és a dir, que permeti la comunicació amb altres sistemes, ja siguin gestors de bases de dades, facilitats aportades pels sistemes operatius, rutines implementades en altres llenguatges, etc.

malauradament el punt c no s'ha trobat en Python, per això s'ha decidit programar amb Matlab.

7.2 Matlab



Per tal d'assolir els objectius que ens hem marcat vàrem utilitzar, bàsicament, l'entorn de treball Matlab R2010a, que es troba disponible per els tres sistemes operatius més comuns: Windows, Mac Os, i Linux.

Es calcula que l'any 2004 més d'un milió d'investigadors i estudiants de diferents parts del món utilitzaven aquest software.

MATLAB és, bàsicament, un llenguatge d'altes prestacions per a la computació en totes aquelles àrees basades en processament de dades, siguin de la índole que siguin. A més del llenguatge, MATLAB s'integra en un entorn d'ús bastant senzill i amigable, una gran quantitat de capacitats de còmput, visualització i programació. Els usos més típics de MATLAB són:

- Anàlisi Matemàtica i Simulacions Numèriques.
- Càlcul Simbòlic.
- Desenvolupament i Test d'Algorismes.
- Modelatge de Sistemes.
- Anàlisi Estadístics i models predictius.
- Gràfics Científics i enginyeria.
- Desenvolupament d'Aplicacions i Productes Finals incloent GUI-Graphical User Interfaces
- etc.

MATLAB és en última instància un entorn de desenvolupament interactiu. L'element bàsic és un array. És a dir, MATLAB entén els vectors i les matrius de la mateixa manera que C o Fortran entenen les variables, sense la necessitat de desenvolupar programació basada en bucles imbricats per realitzar operacions entre arrays.

El Matlab, abreviatura de Matrix Laboratory, és un programa dissenyat per "The Mathworks" destinat al càlcul i l'anàlisi numèric. Permet treballar amb matrius, i vectors de forma ràpida i intuïtiva. A través d'una línia de comandes dóna la possibilitat de realitzar gairebé múltiples operacions amb matrius i vectors de qualsevol mida i tipus: sumes, restes, representacions, aplicacions de filtres, aplicació de mètodes estadístics, umbralitzacions, etc. Tot això fa que sigui molt útil en una gran varietat de camps: tractament i anàlisis d'imatges, adquisició de dades, economia, intel·ligència artificial, etc.

Des de la seva aparició, MATLAB ha anat renovant-se i creixent gràcies a les contribucions dels usuaris. És un producte de l'interès per diferents àrees científic-tècniques. Han sorgit gran quantitat de funcions específiques per a cada àrea, que vénen agrupades en paquets anomenats toolbox. Aquestes toolboxes consisteixen en col·leccions de funcions per MATLAB, funcions desenvolupades per resoldre problemes d'un tipus particular. D'entre aquestes àrea podem citar:

- Processament de Senyal.
- Sistemes de Control.
- Xarxes Neuronals.
- Processament d'Imatges.
- Lògica Difusa.
- Caracterització i Modelatge de Sistemes.
- Economia.
- Equacions Diferencials Multidimensionals.

El sistema complet MATLAB consisteix de 5 blocs diferenciats i que poden funcionar independentment.

- **Entorn de Desenvolupament:** És el conjunt d'eines que li permet usar els arxius i funcions de MATLAB. La majoria d'aquestes eines són interfícies gràfiques per a l'usuari. En concret la de més alt nivell és l'anomenat MATLAB desktop, que inclou el Command Window, al Command History, al Workspace viewer i al Path Browser.

- **Llibreria de funcions:** Aquesta llibreria és una amplíssima col·lecció de algoritmes de còmput de molt diversa complexitat, des de sumes, restes i funcions trigonomètriques fins obtenció de FFTs, etc.

- **Llenguatge de MATLAB:** És un llenguatge de programació d'alt nivell, i l'element bàsic és una matriu. Com a bon llenguatge de programació, inclou sentències de control de flux, estructures, i objectes.

- **Sistema de Gràfics:** Que li permet visualitzar les dades en una gran quantitat de representacions diferents. Gràcies a ordres d'alt nivell es pot visualitzar gràfics bidimensionals i tridimensionals, imatges, animacions, etc. A més inclou ordres de baix nivell que li permeten personalitzar les representacions gràfiques i construir interfícies gràfiques per a usuaris.

- **API - Application Program Interface**-: Aquesta és una llibreria que permetrà escriure-o reutilitzar-les funcions en C o Fortran i fer-les interactuar amb codis escrits en el llenguatge natiu de MATLAB.

Una de les característiques que fan del Matlab una eina molt interessant és la possibilitat de crear noves funcions i afegir-les al programa. Permet la creació d'scripts mitjançant un llenguatge de programació propi anomenat "M-code" o simplement "M". Aquest llenguatge, que és una barreja entre Fortran i C, inclou

totes les funcions del Matlab a més d'estructures típiques dels llenguatges iteratius com ara els bucles. Un tret destacable és que permet la interacció amb programes externs al Matlab, cosa que li permet llançar aplicacions i emmagatzemar-ne els resultats.

Un altre virtut destacable del Matlab és la possibilitat d'afegir-hi diferents "Toolbox". Una Toolbox, caixa d'eines en anglès, és un conjunt d'scripts i funcions externes al programa que s'han desenvolupat per atacar un conjunt de problemes. Solen estar especialitzades en determinats camps com ara la estadística, la biologia, el tractament d'imatges, etc. Podríem dir que són l'equivalent a les llibreries en altres llenguatges de programació. Nosaltres hem utilitzat les següents Toolbox:

- Statistic Toolbox: Eina que permet realitzar diferents càlculs estadístics
- Image Toolbox: Permet analitzar i modificar imatges

El Matlab també inclou la possibilitat de crear senzills entorns gràfics per tal de simplificar l'ús de les funcions. Aquest és un camp que no hem explotat massa en aquest projecte ja que vàrem considerar que el principal objectiu era la investigació i no el desenvolupament d'una aplicació concreta.

7.3 Python

Python és un llenguatge de programació creat per Guido van Rossum a principis dels anys 90 el nom està inspirat en el grup de còmics anglesos "Monty Python". És un llenguatge similar a Perl, però amb una sintaxi molt neta i que afavoreix un codi llegible.

Es tracta d'un llenguatge interpretat o de script amb tipus dinàmics fortament tipat multiplataforma i orientat a objectes.

Es compara habitualment amb TCL, Perl, Scheme, Java i Ruby. En l'actualitat Python es desenvolupa com un projecte de codi obert, administrat per la Python Programari Foundation. Per aquest projecte s'ha emprat la versió 2.5.

Python és considerat com la "oposició lleial" a Perl, llenguatge amb el qual manté una rivalitat amistosa. Els usuaris de Python consideren a aquest molt més net i elegant per a programar. Python permet dividir el programa en mòduls reutilitzables des d'uns altres programes Python. També hi ha mòduls inclosos que proporcionen E/S de fitxers, crides al sistema, sockets i fins a

interfícies a GUI (interfície gràfica amb l'usuari) GTK, Qt, wxWidgets(wxPython), PMW, entre altres.

Python és un llenguatge interpretat, el que estalvia un temps considerable en el desenvolupament del programa, perquè no és necessari compilar ni enllaçar. L'intèrpret es pot utilitzar de manera interactiva, el que facilita experimentar amb característiques del llenguatge, escriure programes d'un sol ús o provar funcions durant el desenvolupament del programa. El principal objectiu que persegueix aquest llenguatge és la facilitat, tant de lectura, com de disseny.

7.4 wxPython

wxPython és un kit d'eines multiplataforma per a la creació d'aplicacions gràfiques. El principal autor de wxPython és Robin Dunn, Amb els desenvolupadors poden crear aplicacions wxPython en diferents sistemes operatius com Windows, Mac i Unix.

wxPython consta dels cinc mòduls bàsics:

Mòdul de controls ofereix els widgets comuns que es troben en les aplicacions gràfiques. Per exemple, un botó, una barra d'eines o un quadern. Els widgets són cridats els controls en el sistema operatiu Windows.

El mòdul de nucli consisteix en classes elementals, que s'utilitzen en el desenvolupament. Aquestes classes inclouen la classe d'objectes, que és la

mare de totes les classes, calibradors, que s'utilitzen per al disseny d'interfícies, Esdeveniments, classes bàsics de geometria com el punt i el rectangle.

La interfície de dispositiu gràfic (GDI) és un conjunt de classes que s'utilitzen per l'elaboració en els widgets. Aquest mòdul conté classes per a la manipulació de fonts, colors, pinzells, llapis o imatges.

Aquestes classes s'utilitzen per al registre, configuració d'aplicacions, la configuració del sistema, en col·laboració amb la **El mòdul Miscel·lània** conta pantalla o la palanca de control.

El mòdul de Windows consta de diverses finestres, que formen una aplicació. Grup, de diàleg, marc o finestra de desplaçament.

7.5 PIL

El Python Imaging Library (PIL) li permet crear, modificar i convertir arxius d'imatge en una àmplia varietat de formats utilitzant el llenguatge Python.

La llibreria d'imatges de Python afegeix capacitats de processament d'imatges a l'intèrpret Python. Aquesta biblioteca proporciona una àmplia compatibilitat de formats d'arxiu, una representació interna eficient, i capacitats d'imatge bastant potents de processament.

La biblioteca d'imatges del nucli està dissenyada per a un ràpid accés a les dades emmagatzemades en un píxel. S'ha de proporcionar una base sòlida per a una eina de processament d'imatges en general. Fem una ullada a alguns usos possibles d'aquesta biblioteca:

- **Arxiu d'Imatges**

La Biblioteca d'imatges de Python és ideal per arxius d'imatge i processament per moltes aplicacions. Es pot utilitzar la biblioteca per crear miniatures, convertir entre formats d'arxiu, imprimir les imatges, etc

La versió actual identifica i llegeix un gran nombre de formats. Escriure és intencionalment restringit a l'intercanvi més comunament usat i formats de presentació.

- **Visualització d'imatges**

La versió actual inclou Tk PhotoImage i interfícies BitmapImage, així com una interfície de Windows DIB que es pot utilitzar amb PythonWin. Per X i pantalles de Mac, pot utilitzar la biblioteca img de Jack Jansen.

Per a la depuració, també hi ha un mètode de demostració en la versió de Unix que es requereix la llibreria xv per mostrar la imatge.

- **processament d'Imatges**

La biblioteca conté algunes funcions de processament d'imatge bàsiques, incloent operacions de punt, filtrat amb un conjunt de nuclis de convolució incorporats, i les conversions d'espai de color.

La biblioteca també pot redimensionar tant els d'imatge, rotació i arbitràries transformacions afins.

7.6 GANTT Project

Gantt és un diagrama o gràfic de barres que es fa servir quan és necessari representar l'execució o la producció total. Aquesta mostra l'ocurrència d'activitats en paral·lel o en sèrie en un determinat període de temps. Tenen per objecte controlar l'execució simultània de diverses activitats que es realitzen coordinadament.

Aquest va ser desenvolupat per Henry L. Gantt el 1917 i és una senzilla eina de gràfics de temps, ja que són fàcils d'aprendre, llegir i escriure. Aquests resulten bastant eficaços per la planificació i l'avaluació de l'avenç dels projectes.

Un gràfic de Gantt és un senzill gràfic de barres. Cada barra simbolitza una tasca del projecte. On l'eix horitzontal representa el temps. Com aquests gràfics es fan servir per encadenar tasques entre si, l'eix horitzontal hauria d'incloure dates. Verticalment, i en la columna esquerra, s'ofereix una relació de les tasques.

7.7 Argouml

ArgoUML és una eina utilitzada en el modelatge de sistemes, mitjançant la qual es realitzen dissenys en UML ("Unified Markup Language") portats a terme en l'anàlisi i pre-disseny de Sistemes de Programari.

7.8 GUIDE - interfície gràfica d'usuari en Matlab

GUIDE és un entorn de programació visual disponible en MATLAB per realitzar i executar programes que necessitin ingrés continu de dades. Té les característiques bàsiques de tots els programes visuals com Visual Basic o Visual C ++.

El llenguatge més habitual per a crear GUI-s és Java, ja que té l'enorme avantatge de funcionar en qualsevol màquina, però Java és molt lent per fer càlculs eficientment, i és aquí on Matlab és més poderós. D'altra banda, les GUI-s creades amb Matlab poden ser lliurades a l'ordinador d'un client (qui possiblement no tingui més que un navegador) i ser executades a l'ordinador de qui va crear la interfície en Matlab (i que per descomptat té un Matlab funcionant), de manera que l'avantatge relatiu de Java està parcialment oferta també pel Matlab.

Les GUI-s són eines molt útils per lliurar aplicacions a aquelles persones que no saben prou de programació i que volen beneficiar-se dels avantatges d'un programa.

7.9 Elecció del sistema operatiu

Tal i com s'indica en l'apartat anterior, el Matlab es troba disponible per els sistemes operatius més usats. Això ens donava llibertat a l'hora d'escollir quin utilitzaríem per desenvolupar el nostre projecte, en aquest cas s'ha fet en Windows.

7.9 Elecció de la plataforma de desenvolupament

Després de definir els passos que calia seguir per assolir els objectius proposats vàrem que necessitaríem utilitzar funcions del Matlab com ara detectors de regions "regionprop", que es una descripció bàsica de regions, a través de propietats bàsiques com: àrea, nombre de euler, mínim rectangle que envolta la figura, etc, i amb aquesta propietat que no es troba en Python s'ha decidit la primera part del projecte.

Vàrem observar que aquest programa (Matlab) no es gratuït, però esta disponible als ordinadors de la politècnica superior, i vam decidir desenvolupar el projecte dins d'un entorn Windows perquè disposàvem d'una llicencia en aquest entorn.

Més sobre regionprop:

Les propietats que es volen calcular es van posant una darrera l'altra seguides per una coma. Les propietats disponibles són

- area - Calcula l'àrea en píxels quadrats de la regió.
- BoundingBox - Calcula la posició i dimensions del mínim rectangle que envolta la regió.
- centroid - Calcula la posició del centroide de la regió.
- convexhull - Retorna una matriu amb la posició dels píxels que defineixen el casc convex que envolta la regió.
- conveximage - Imatge binària amb la forma del casc convex.
- eccentricity - Nombre escalar que l'excentricitat de la regió.
- eulernumber - Calcula el nombre d'Euler de la regió.
- mayoraxislength - Calcula la longitud de l'eix major de la regió.
- minoraxislength - Calcula la longitud de l'eix menor de la regió.

La funció **regionprops** retorna una matriu que conté l'estructura de les mesures desitjades per a cada objecte.

La sintaxi de la llista separada per comes de les estructures ofereix una manera fàcil de convertir una àrea determinada d'una àmplia estructura en un vector.

En el programari s'ha fet servir les dues propietats:

- Area
- BoundingBox

7.10 Detectors de regions

En el camp de la visió artificial, el **'reconeixement de regions'** es refereix a les tècniques amb l'objectiu de detectar punts o regions més clares o més fosques de la imatge. Hi ha dues classes principals de detectors de regions: mètodes diferencials i mètodes basats en extrems locals. Aquests detectors també es denominen detectors de punts interessants, o detectors de regions interessants.

L'estudi i desenvolupament d'aquests detectors és important per diverses raons. La principal és donar informació complementària sobre regions que no es pot obtenir mitjançant detectors de vores o detectors de cantonades. Els detectors de regions s'usen com a pas previ per al reconeixement d'objectes o seguiment d'objectes. Un altre ús habitual d'aquests detectors té a veure amb l'anàlisi de textures i el seu reconeixement. Recentment, els descriptors de regions han començat a usar-se per llocs d'interès per informar de la presència de determinats objectes en una imatge.

Aquestes tècniques, en combinació amb altres, tenen ja aplicacions d'ús més quotidià: per exemple per a programari de dispositius tàctils, facilitats de detecció de rostres i somriures en càmeres de fotos, sistemes de vigilància i seguretat, o per analitzar imatges mèdiques (Diagnòstic Assistit per Ordinador).

Capítol 8: Anàlisi i disseny del sistema

Un requeriment és una característica de disseny, atribut, o comportament d'un sistema. Amb les necessitats d'un sistema, s'acorda un pacte establert entre les coses fora del sistema i el sistema per si mateix, quan es declara que s'espera que el sistema faci. En la major part no es cuida com el sistema ho fa, només es cuida que és el que fa. Quan es construeix un sistema és important iniciar amb acords sobre que és el que farà el sistema, encara que es podrà desenvolupar l'enteniment d'aquests requeriments acord iterativa i incrementalment es desenvolupi el sistema.

En aquest capítol descrivim el que fa un sistema des del punt de vista d'un usuari fent servir UML ("Unified Modeling Language").

8.1 Disseny general

En aquest apartat s'expliquen les eines que es faran servir per poder desenvolupar les diferents parts del disseny d'aquest programa i es mostraran els casos que poden tenir. Per poder-ho fer millor, el primer que s'utilitzarà són els diagrames de casos d'us.

A continuació es mostra l'esquema general del projecte en la figura 13a.

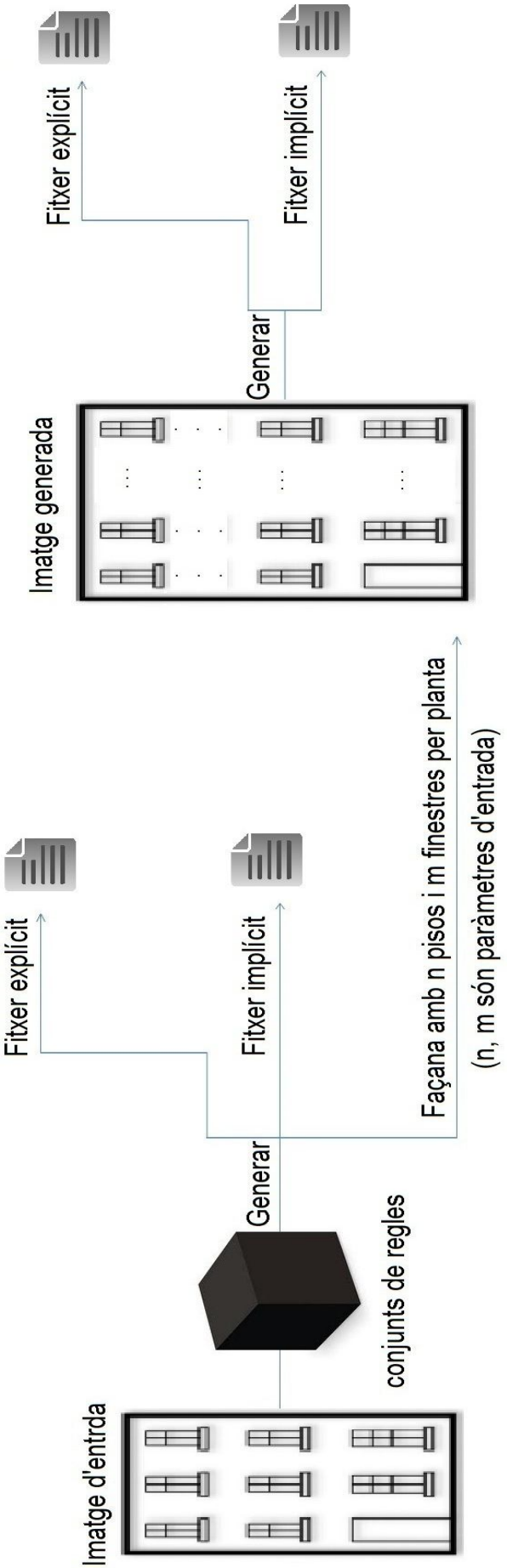


Figura13a: l'esquema general de l'aplicació

8.2 Identificació dels actors

A continuació s'identifiquen els actors de l'aplicació. Un actor és qualsevol cosa que intercanvia dades amb el sistema. És una entitat externa (persona, dispositiu, procés, subsistema, temps, ..). Un actor sempre tindrà interacció amb el sistema ja sigui inicialitzant el cas d'ús o intercanviant informació.

Els actors:

- no són part del sistema que es construeix;
- entren informació al sistema;
- reben informació del sistema.

En el programa no hi ha cap tipus de manteniment, es tracta d'un programa en el que no existeix la distinció entre els possibles usuaris que el puguin fer servir, o que interactuarà en tot moment amb el sistema. Veure Figura 14.

Figura 14: Identificació dels actors

8.3 Diagrama de cas d'us general

Els diagrames de casos d'ús documenten el comportament d'un sistema des del punt de vista de l'usuari. Per tant els casos d'ús determinen els requisits funcionals del sistema, és a dir, representen les funcions que un sistema pot executar. El seu avantatge principal és la facilitat per interpretar-los, el que fa que siguin especialment útils en la comunicació amb el client.

Un cas d'ús descriu el comportament (funcionalitats) d'un sistema quan interactua amb un usuari extern (actor). Un cas d'ús representa un conjunt d'interaccions entre un o més actors i el sistema. Els casos d'ús defineixen el comportament d'un sistema des del punt de vista dels actors.

Els casos d'ús:

- representen els requeriments funcionals d'un sistema;

- descriuen què fa un sistema, no com ho fa;
- dirigeixen tot el procés de desenvolupament d'un sistema.

A la figura 15 es mostra el diagrama de cas d'us de context general del projecte.

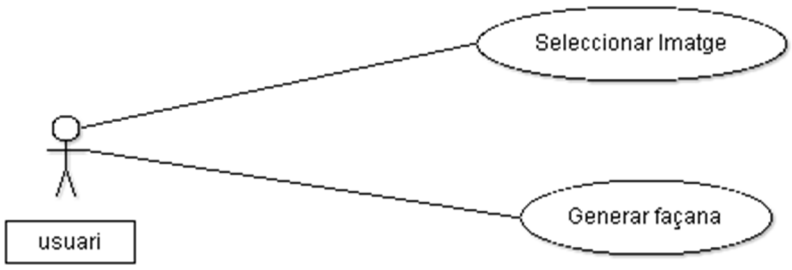


Figura 15: Diagrama de cas d'us general

Casos d'ús:

- Seleccionar Imatge
- Guardar arxiu .txt explícit de la imatge
- Guardar arxiu .txt implícit de la imatge
- Omplir paràmetres
- Generar Façana
- Guardar Imatge Façana generada
- Guardar arxiu .txt explícit de la Façana generada
- Guardar arxiu .txt implícit de la Façana generada

Fitxa	Generar arxiu .txt explícit de la imatge
Descripció	L'usuari entra una imatge per generar un arxiu .txt explícit
Actor	Usuari
Pre-condició	La imatge ha de ser regular
Flux Principal	1.Seleccionar imatge 2.Generar arxiu .txt explicit
Subfluxos	Es guarda totes les finestres i portes en una carpeta
Flux alternatiu	Si la imatge no es regular dóna error
Pos-Condició	Generar un arxiu .txt explicit i guardat on hem indicat.

Fitxa	Generar arxiu .txt implícit de la imatge
Descripció	L'usuari entra una imatge per generar un arxiu .txt implícit
Actor	Usuari
Pre-condició	La imatge ha de ser regular
Flux Principal	1.Seleccionar imatge 2.Generar arxiu .txt implícit
Subfluxos	Es guarden totes les finestres i portes en una carpeta
Flux alternatiu	Si la imatge no es regular dóna error
Pos-Condició	Generar un arxiu .txt implícit i guardat on hem indicat.

Fitxa	Generar Façanes a partir de paràmetres d'entrada
Descripció	L'usuari entra una imatge per generar Façanes
Actor	Usuari
Pre-condició	La imatge ha de ser regular El número de pisos ha de ser més gran o igual a 0 (0 indica que només hi ha la planta baixa) El número de finestres per columnes ha de ser més gran o igual a 2.
Flux Principal	1.Seleccionar imatge 2.Entrar número de pisos 3.Entrar número de finestres per pis 4.Escollir opció 4.1 Façana amb repeticions de finestres i portes 4.2 Façana amb repeticions de finestres 5. Generar Façana 6. Guardar Façana generada
Subfluxos	Cap
Flux alternatiu	Si la imatge no és regular dóna error Si els paràmetres no compleixin la condició establerta dóna error.
Pos-Condició	Imatge mostrada i guardada.

Fitxa	Generar arxiu .txt explícit de la façana generada
Descripció	L'usuari entra una imatge per guardar un arxiu explícit de la façana generada
Actor	Usuari
Pre-condició	La imatge ha de ser regular El número de pisos ha de ser més gran o igual a 0 (0 indica que només hi ha la planta baixa) El número de finestres per columnes ha de ser més gran o igual a 2.
Flux Principal	1.Seleccionar imatge 2.Entrar número de pisos 3.Entrar número de finestres per pis 4.Escollir opció 4.1 Façana amb repeticions de finestres i portes 4.2 Façana amb repeticions de finestres 5. Generar Façana 6. Guardar arxiu explícit de la façana generada
Subfluxos	Cap
Flux alternatiu	Si la imatge no és regular dóna error Si els paràmetres no compleixin la condició establerta dóna error.
Pos-Condició	Arxiu explícit generat i guardat.

Fitxa	Generar arxiu .txt implícit de la façana generada
Descripció	L'usuari entra una imatge per guardar un arxiu implícit de la façana generada
Actor	Usuari
Pre-condició	La imatge ha de ser regular El número de pisos ha de ser més gran o igual a 1 El número de finestres per columnes ha de ser més gran o igual a 2.
Flux Principal	1.Seleccionar imatge 2.Entrar número de pisos 3.Entrar número de finestres per pis 4.Escollir opció 4.1 Façana amb repeticions de finestres i portes 4.2 Façana amb repeticions de finestres 5. Generar Façana 6. Guardar arxiu implícit de la façana generada
Subfluxos	Cap
Flux alternatiu	Si la imatge no és regular dóna error Si els paràmetres no compleixin la condició establerta dóna error.
Pos-Condició	Arxiu implícit generat i guardat.

8.4 Diagrama de cas d'us de modificació de paràmetres

El diagrama de la figura 16 mostra els diferents paràmetres que l'usuari pot canviar per obtenir diferents façanes, i a més a més es pot generar una altra façana sense canviar els paràmetres. Això és degut a que les finestres es trien de forma aleatòria.

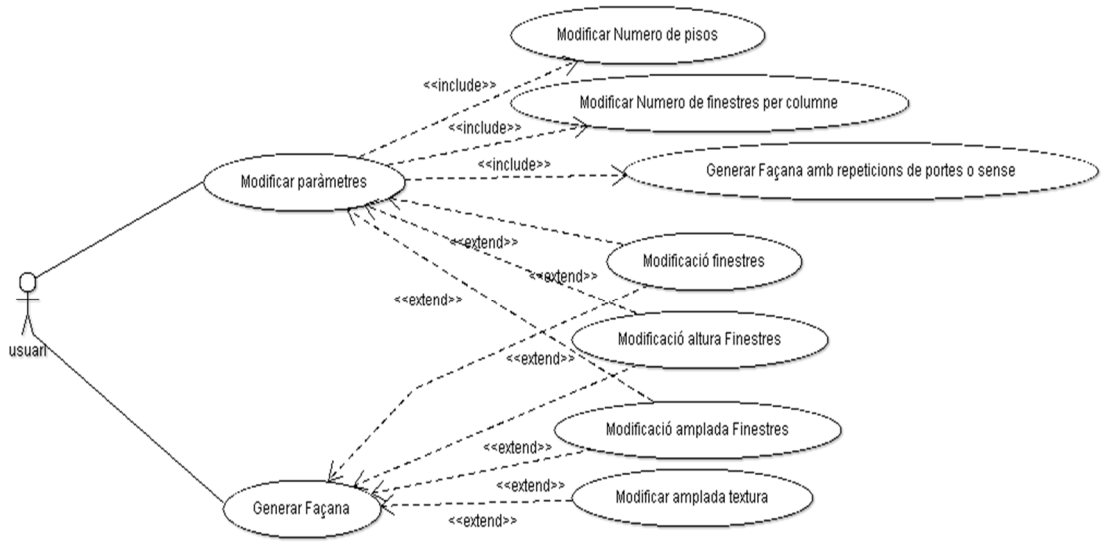


Figura 16: Diagrama de cas d'us de modificació de paràmetres

Fitxa	Modificació de paràmetres
Descripció	L'usuari entra nous paràmetres
Actor	Usuari
Pre-condició	El número de pisos ha de ser més gran o igual a 1 El número de finestres per columnes ha de ser més gran o igual a 2.
Flux Principal	1. Modificar Número de pisos 2. Modificar Número de finestres per pis 3. Modificar opció: 3.1 Repeticions de finestres i portes 3.2 Repeticions de finestres 4. prem el botó GENERAR
Subfluxos	Cap
Flux alternatiu	Si els paràmetres no compleixin la condició establerta dóna error.
Pos-Condició	Façana mostrada

Fitxa	Generar façana
Descripció	L'usuari genera façana modificat
Actor	Usuari
Pre-condició	El número de pisos ha de ser més gran o igual a 1 El número de finestres per columnes ha de ser més gran o igual a 2.
Flux Principal	1. prem el boto GENERAR
Subfluxos	1.Modificar les posicions de les finestres 2.Modificar altura finestres 3.Modificar amplada finestres 4.Modificar amplada textura
Flux alternatiu	Si els paràmetres no compleixin la condició establerta dóna error.
Pos-Condició	Façana mostrada

8.5 Diagrama de activitat

El diagrama d'activitat es centra en el flux d'activitats involucrades en un procés, generalment dins del marc d'un o diversos casos d'ús. Un diagrama d'activitats mostra en quin ordre s'executen les parts del procés i com depenen unes de les altres.

El diagrama d'activitat no proporciona informació del comportament d'un objecte o de les col·laboracions entre objectes.

En un diagrama d'activitat, el procés comença a partir del cercle negre d'inici situat a la part superior o esquerra del diagrama i acaba al cercle blanc/negre de final situat a la part inferior o dreta del diagrama. Les activitats s'indiquen amb rectangles arrodonits.
Els diagrames d'activitat es poden subdividir en *carrers* (swimlanes) per a mostrar el responsable (actor, objecte, unitat organitzacional, cas d´ús, ...) encarregat de l'activitat.
De cada activitat se'n deriva una transició que connecta amb la següent activitat.

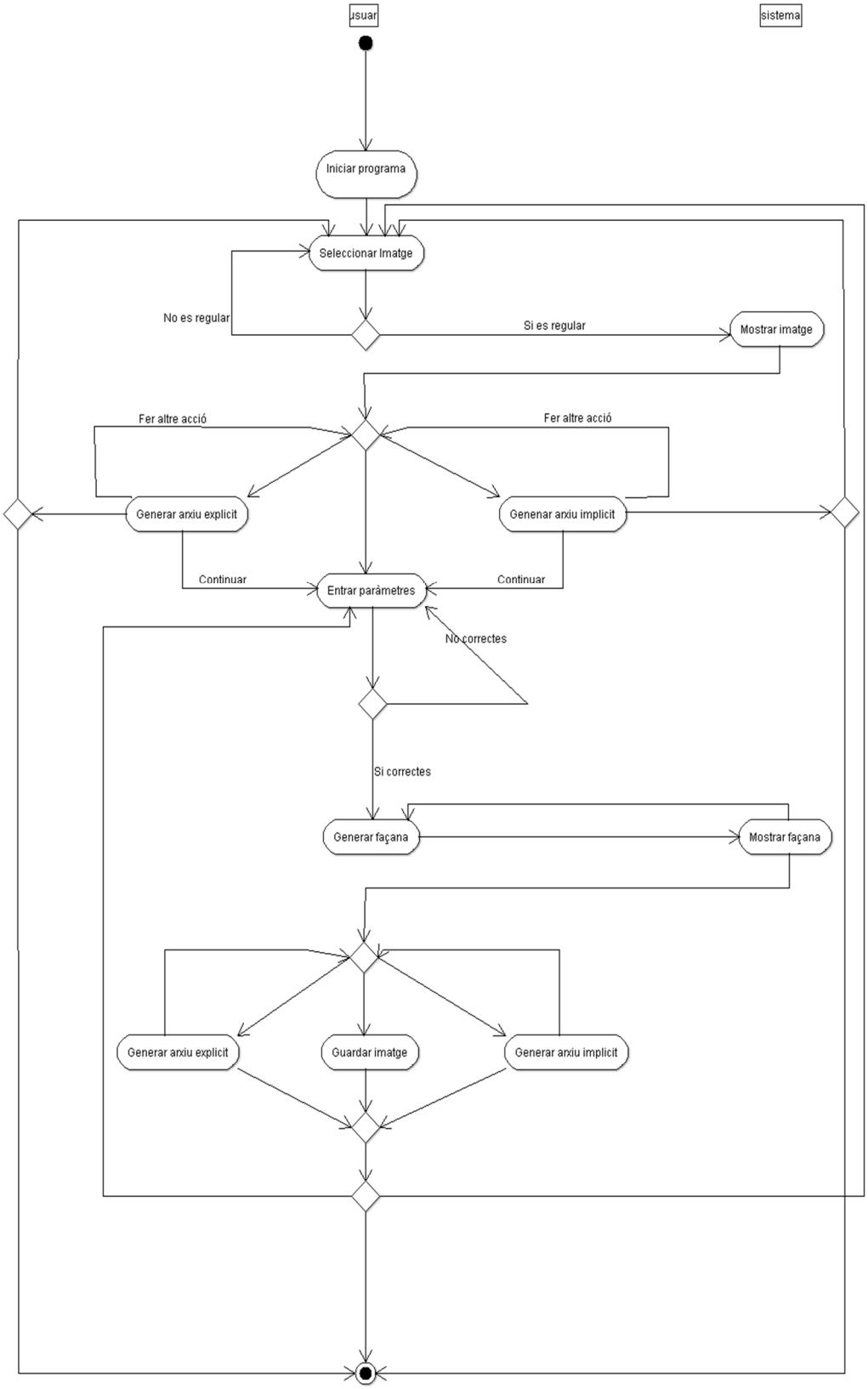


Figura 17: Diagrama d'activitat

El flux d'aquest projecte (veure figura 17) comença amb l'usuari que, després de seleccionar la façana, té tres opcions amb aquesta imatge de entrada, que són:

1. Guardar un arxiu explícit de la façana
2. Guardar un arxiu implícit de la façana
3. Generar Façana: Una vegada l'usuari ha omplert els paràmetres i polsat el boto generar, l'usuari tindrà quatre opcions amb aquesta imatge generada:
    - guardar la imatge
    - guardar l'arxiu explícit de la imatge
    - guardar l'arxiu implícit de la imatge
    - Modificar els paràmetres, en el cas que l'usuari vulgui modificar la imatge generada, tindrà dos opcions:
        - Prémer el botó generar, perquè el que fa es col·locar les finestres de forma aleatòria. Això vol dir que cada vegada que polsem el botó generar sortirà una altra façana amb finestres diferents, però amb els mateixos paràmetres
        - Modificar els paràmetres.

La façana ha de ser regular, sinó sortirà un error i li demanarà seleccionar una altra imatge. El sistema mostrà la imatge cada vegada quan s'ha seleccionat una regular, o també quan generarà una altra.

8.6 Processament d'imatges

La visió per computador actualment comprèn tant l'obtenció com la caracterització i interpretació de les imatges. Això suposa algorismes de molt diversos tipus i complexitats. En un sistema de visió per computador actual es poden distingir sis etapes o fases (figura 18):

- **Captació**: És el procés a través del qual s'obté una imatge visual.
- **Preprocessament**: Inclou tècniques com ara la reducció de soroll i realç de detalls.
- **Segmentació**: És el procés que divideix una imatge en objectes que siguin del nostre interès.
- **Descripció**: És el procés mitjançant el qual s'obtenen característiques convenients per diferenciar un tipus d'objecte d'un altre, per exemple mida i forma.
- **Reconeixement**: És el procés que identifica els objectes d'una escena. Per exemple les diferents tipus de peces en un tauler de escacs.
- **Interpretació**: És el procés que associa un significat a un conjunt de objectes reconeguts.

Figura 18: Esquema general d'un sistema de visió per computador.

Això suposa diferents tipus de processament en funció del nivell en què ens moguem:

Visió de baix nivell: comprèn la captació i el preprocessament. Executa algorismes típicament de filtrat, restauració de la imatge, realç, extracció de contorns, etc.

Visió de nivell intermedi: comprèn la segmentació, descripció i reconeixement, amb algoritmes típicament d'extracció de característiques, reconeixement de forma i etiquetatge d'aquestes.

Visió d'alt nivell: comprèn la fase d'interpretació, normalment aquests algorismes es refereixen a la interpretació de les dades generalment mitjançant procediments típics de la Intel·ligència Artificial per a accés a bases de dades, cerques, raonaments aproximats, etc.

8.7 Segmentació

La segmentació és un procés que consisteix a dividir una imatge digital en regions homogènies respecte a una o més característiques (com ara la brillantor o el color) amb la finalitat de facilitar una posterior anàlisi o reconeixement automàtic. Localitzar la cara d'una persona dins de la imatge d'una fotografia o trobar els límits d'una paraula dins d'una imatge d'un text, constitueixen exemples de problemes de segmentació (figura 19).

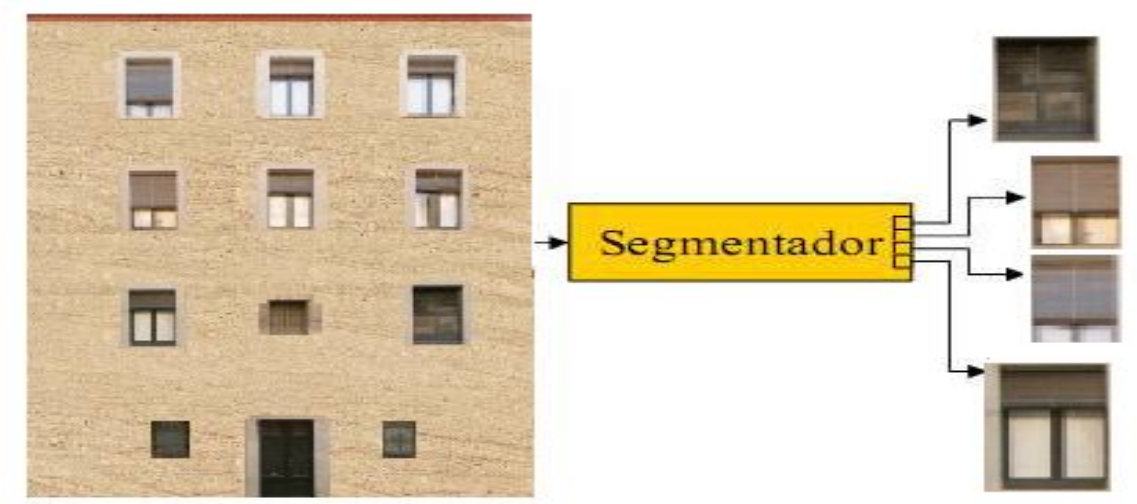


Figura 19: Exemple de segmentació en què s'extreuen de la imatge finestres i portes.

8.7.1 Conceptes bàsics sobre segmentació

La segmentació s'ha de veure com un procés en el qual, a partir d'una imatge d'entrada, es produeix una altra en la qual cada píxel té associada una etiqueta distintiva de l'objecte a què pertany. Així, un cop segmentada una imatge, es podria formar una llista d'objectes consistents en les agrupacions dels píxels que tinguin la mateixa etiqueta.

La segmentació acaba quan els objectes extrets de la imatge es corresponen unívocament amb les diferents regions disjuntes a localitzar a la mateixa. En aquest cas es parla de segmentació completa de la imatge i en el cas contrari, de segmentació parcial. En una escena complexa, el resultat de la segmentació podria ser un conjunt de regions homogènies superposades i en aquest cas, la imatge parcialment segmentada haurà de ser sotmesa després a un tractament posterior amb la finalitat d'aconseguir una segmentació completa.

8.7.2 La textura

Intuïtivament la textura d'un objecte dins d'una imatge és el conjunt de formes que s'aprecia sobre la superfície i que el dota de cert grau de regularitat. Una definició clàssica de textura és la següent: "un o més patrons locals que es repeteixen de manera periòdica".

Hi ha dos enfocaments per definir una textura: un descendent ("top-down") i un altre ascendent ("bottom-up"). L'enfocament descendent es basa en l'existència d'un element bàsic de textura, anomenat texel, i en una regla de formació. Aquesta regla defineix com i on es situen aquests elements bàsics. Aquest enfocament funciona bé quan la textura és bastant regular, per exemple en la imatge d'una paret de maons. D'altra banda, l'enfocament ascendent assumeix

que la textura és una propietat que es pot derivar de estadístiques (com la mitjana i la variància) de petits grups de píxels. Aquest enfocament funciona bé per textures on resulta difícil veure els components individuals, com per exemple una textura de l'herba o de pedra. No obstant això, la línia divisòria entre els dos enfocaments no és clara.

8.7.3 El contorn

El contorn d'un objecte en una imatge digital correspon al mínim conjunt de píxels que separa aquest objecte del fons o background de la imatge. Normalment aquests contorns es corresponen amb els punts on es produeixen discontinuïtats en els valors de píxels adjacents (canvis en el matís o la brillantor) o amb els punts on canvia un patró que es repeteix (canvis de textura).

8.7.4 Segmentació basada en L'umbralització

La umbralització és un procés que permet convertir una imatge de nivells de gris o de color en una imatge binària, de manera que els objectes d'interès s'etiquetin amb un valor diferent al dels píxels del fons.

La umbralització és una tècnica de segmentació ràpida, que té un cost computacional baix i que pot ser realitzada en temps real durant la captura de la imatge usant un ordinador personal de propòsit general.

Viem un exemple de segmentació per llindar en la figura 20.

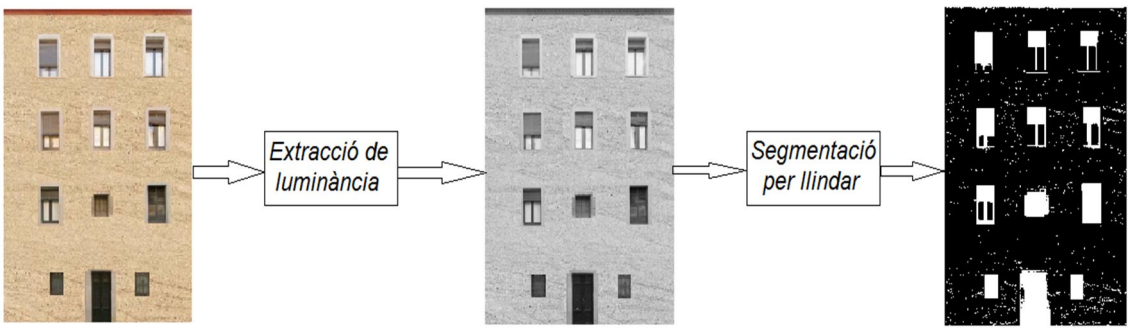


Figura 20: Extracció de components de la façana mitjançant segmentació per llindar.

8.8 Variables i funcions de la classe

Una llibreria compartida és una col·lecció de funcions executables a punt per ser utilitzades en una o més aplicacions. En aquest sentit, MATLAB permet utilitzar llibreries externes que s'hagin generada en sistemes MS-Windows i Linux.

Tot seguit s'expliquen les funcions del programa. Explicarem tots els atributs, mètodes i el que faci necessari per entendre-les.

ProcessarFaçana
Img: matriu n*m*3 (3: RGB)
BinaritzarImatge TreureSoroll DetectarFinestres EliminarAreasIndesitjables DefinirNumeroPisos DelimitarPisos OrdenarPerAmplada TrobarMinMaxFinestresPerColumna DelimitarTextura PosicioPorta GenerarFitxerExplicitOriginal GeneralFitxerImplicitOriginal GenerarFaçana GenerarFaçanaFinestresExplicit GenerarFaçanaFinestresImplicit DibuixarTerrat CopiarSubImatge PosarDades ConvertirPixelMetres

Figura 21: Classe ProcessarFaçana

Atributs:

img: Una imatge que els píxels són especificats per 3 valors, un per a cada component de color (vermell, verd i blau) de cada píxel. En Matlab, una imatge RGB és representada per un array mxnx3.

Mètodes:

- **BinaritzarImatge**

Converteix la imatge en escala de grisos. La imatge de sortida substitueix a tots els píxels de la imatge d'entrada amb luminància major que el nivell amb el

valor 1 (blanc) i substitueix tots els altres píxels amb el valor 0 (negre) (veure figura 22).

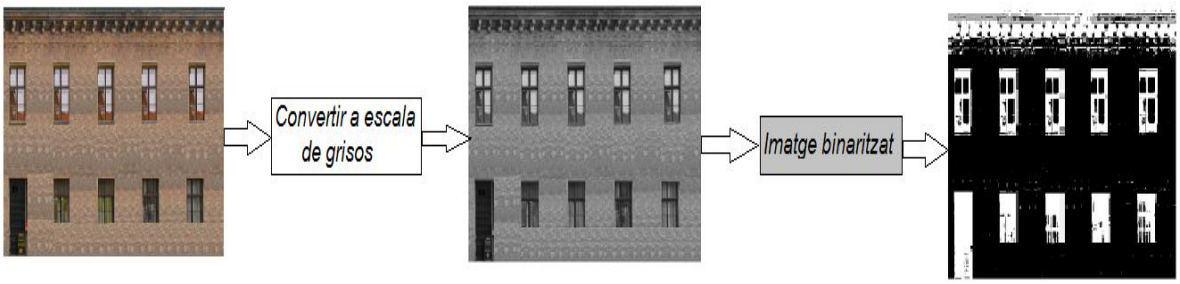


Figura 22: Binarzació d'imatge d'escala de grisos

- **TreureSoroll**

Eliminar a partir de la imatge binària tots els components connectats (objectes) que tenen menys d'un número de píxels donat com a valor d'entrada del mètode, produint una altre imatge binària de sortida (figura 23).

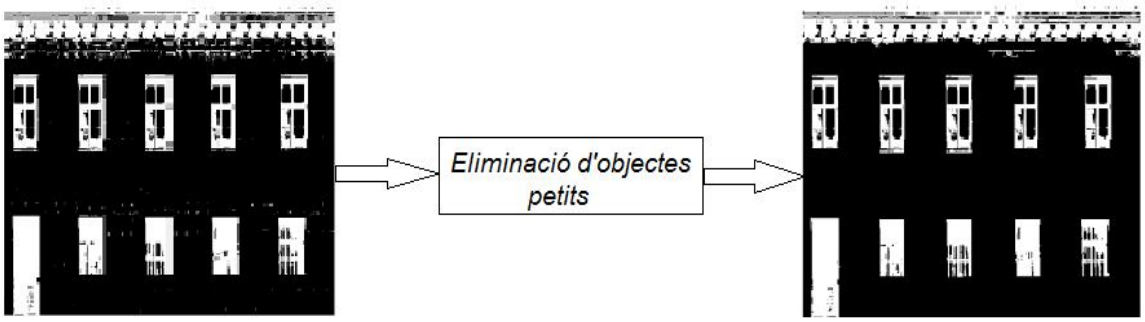


Figura 23: imatge binària extret els components més petits

- **DetectarFinestres**

Funció que a partir de la imatge binària, primerament calcula els components connectats per imatges. Retorna una matriu de la mateixa mida que la imatge d'entrada que conté les etiquetes per als objectes connectats.

La funció retorna una matriu que conté [y, x, w, h] per cada objecte trobat. Després s'ha fet servir una funció del Matlab anomenda "Rectangle" per dibuixar el rectangle des del punt x, y, que té una amplada de w i una alçada h (figura 24).

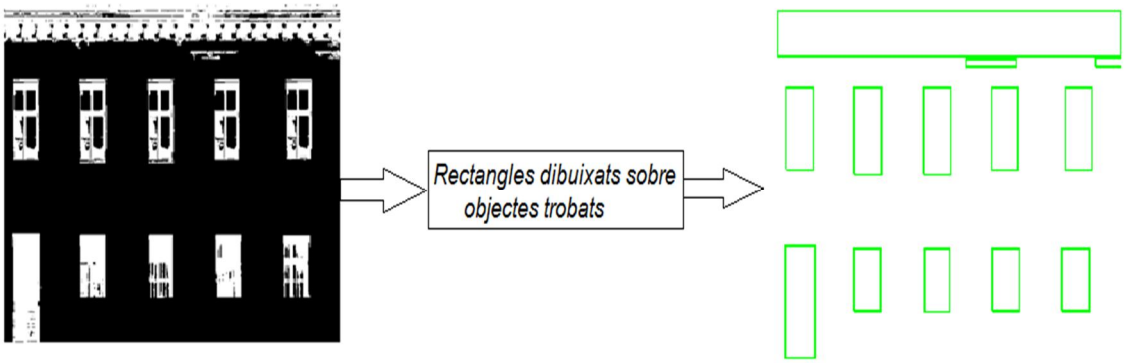


Figura 24: dibuixar rectangles sobre objectes trobats

- **EliminarAreasIndesitjables**

Funció que retorna una imatge a partir de la imatge binaritzada, es a dir, elimina els components més petits que no són ni finestres ni portes. En la funció anterior s'han eliminat components molt petits, però s'han deixat altres: en la figura 25 estan requadrats en vermell els que no s'han de mostrar en la imatge. En aquesta funció s'ha aprofitat a eliminar més objectes fent comparacions entre àrees i eliminar les àrees menors a 6 vegades l'àrea promig de la finestra.

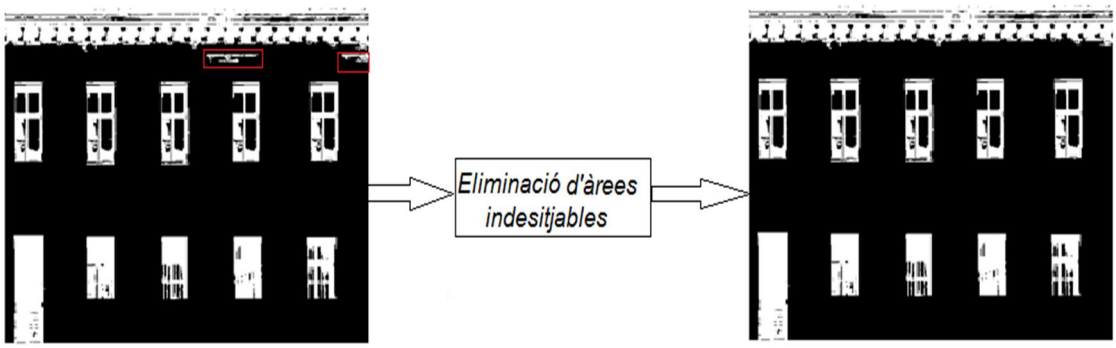


Figura 25: Eliminació d'àrees indesitjables

- **DefinirNumeroPisos**

Funció que retorna el número de pisos i una taula que conté el número de finestres en cada pis a partir de la matriu que conté [y, x, w, h] trobada en la funció **DetectarFinestres.**

Aquesta funció ordena les components per alçada x. Així queda al final, en cada alçada, varies components que són les finestres per pis. En la figura 26 veiem la sortida de la funció i observem que en la façana hi ha un pis i la planta baixa.

Posició	Y	X	W	H	
1	17.5000	100.5000	46.0000	108.0000	
2	132.5000	100.5000	48.0000	113.0000	
3	251.5000	100.5000	47.0000	113.0000	Pis1
4	369.5000	100.5000	43.0000	108.0000	
5	494.5000	100.5000	45.0000	108.0000	
6	15.5000	309.5000	51.0000	**146.0000**	
7	133.5000	311.5000	45.0000	83.0000	Planta baixa
8	252.5000	311.5000	44.0000	84.0000	
9	370.5000	311.5000	46.0000	83.0000	
10	489.5000	312.5000	47.0000	82.0000	

Figura 26: matriu de sortida de la funció DifinirNumeroPisos

Si ens fixem en la segona columna dels objectes trobats, veurem que estan ordenats per alçada x. La primera observació és el número de x que coincideix, i que són 5 finestres pel primer pis i altres 5 per la planta baixa.

A continuació es mostra el pseudocodi d'aquest algorisme.

```
función DefinirNumeroPisos(finestres_ordenades: matriu que conté les posicions [y, x, w, h] de les finestres i ordenada per x)
{PRE:: taula de 4 columnes [y, x, w, h] i n files que conté les posicions de les Componentes (y: la posició on comença horitzontalment l'element, w: amplada de l'element, x: posició on comença verticalment l'element, h: alçada de l'element)}
{POST::retorna el número de pisos de la façana, y una taula que conté el número de components en cada pis}

   pisos = 1;
   numElem = size(finestres_ordenades);
   per i de 1 fins numElem fer
      h1 = finestres_ordenades[i][2];      % la posició x on comença verticalment la
                                           % finestra
      h2 = finestres_ordenades[i+1][2];    % la posició x on comença verticalment la
                                           % següent finestra

   si abs(h1-h2)<=20 llavors               % si la diferència verticalment entre finestres
      j = j + 1;                           % és més gran que 20 píxels, és a dir que
      t_pisos = j;                         % passem al següent pis.
   altrament                               % altrament comptem una finestres més en el
      pisos = pisos + 1;                   % pis
      t_pisos[pisos] = 1;
      j = 1;
   fsi
retorna <pisos, t_pisos>
```

```
    fper
ffunció
```

- **DelimitarPisos**

Funció que retorna a partir de la matriu dels objectes ordenats per alçada x, una taula que conté els delimitacions dels pisos, on comencen i on acaben (figura 27).

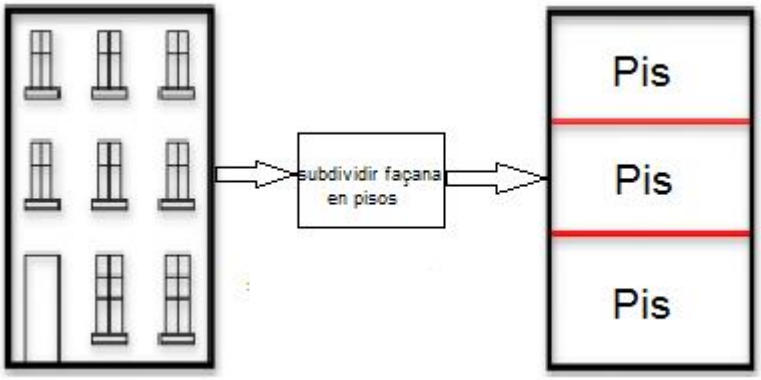


Figura 27: subdivisió de façana en pisos

En la figura 28 es mostra l'algorisme de com detectar un pis.

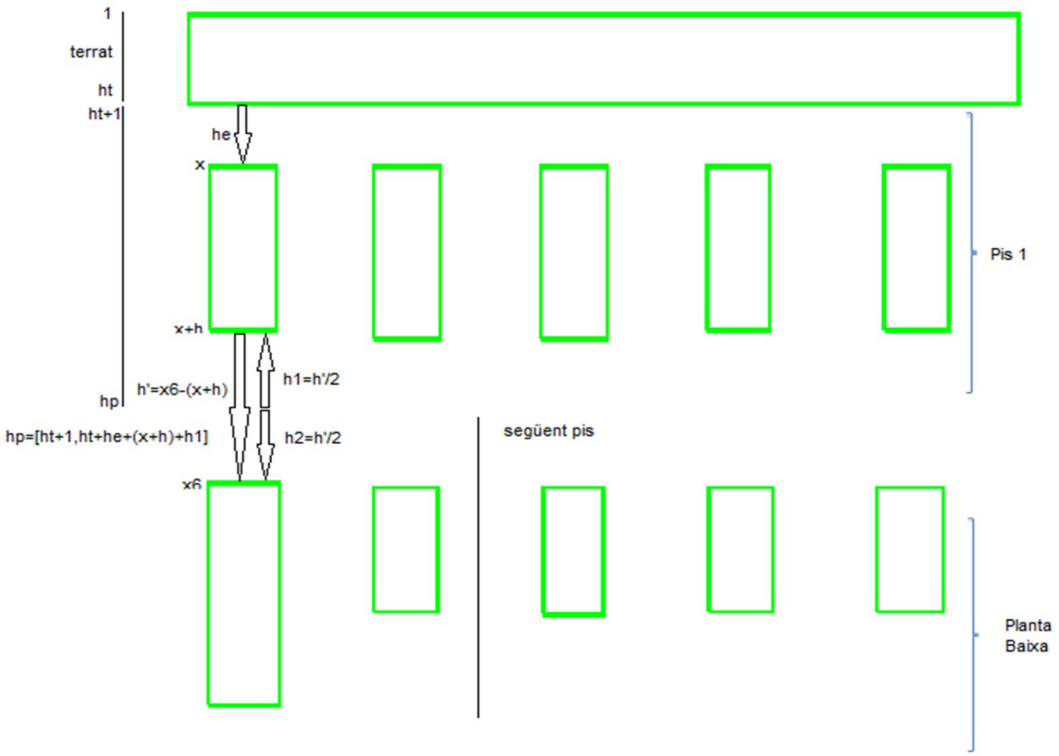


Figura 28: mètodes utilitzat per delimitar pisos

A continuació es mostra el pseudocodi d'aquest algorisme.

```
función DelimitarPisos(finestres_ord: matriu que conté les posicions  [y, x, w, h] de les
finestra i ordenada per x, pisos: número de pisos, t_pisos: taula que conté el número
d'elements per pis)
{PRE:: finestres_ord: matriu que conté les posicions  [y, x, w, h] de les finestra i
ordenada per x, pisos: número de pisos, t_pisos: taula que conté el número
d'elements per pis }
{POST::retorna una taula que conté les posicions on comença i on acaba cada pis}
   h_inici = PassarTerrat();  %funció que retorna la posició on comença l'últim pis
                              % després de passar terrat
   h_seg = 1;
   h_ant = h_inici;
   per i de 1 fins pisos fer
      h_seg = h_seg + t_pisos[i];
      h1 = finestres_ord[h_seg -1][2] + finestres_ord[h_seg -1][4]; % h1 es la posició on
      h2 = finestres_ord[h_seg][2];                % acaba la finestra verticalment sumant
                                                   % l'altura on comença i llargada
      mig = abs((h2-h1)/2);     % la distancia on acaba la primera finestra amb on comença
      p[2*i – 1] = h_ant;       % la següent i dividir por 2 ens dona la distancia del mig
      p[2*i] =  finestres_ord[h_seg – 1][2] + finestres_ord[h_seg -1][4] + mig
      h_ant = p[2*i] + 1;
   fper

retorna p
ffunció
```

- **OrdenarPerAmplada**

Funció que retorna una taula dels components trobats, ordenats per amplada, ja que aquesta taula de entrada ja està ordenada per altura en la funció **DefinirNumeroPisos.**

- **TrobarMinMaxFinestresPerColumna**

Retorna una taula que conté el mínim on comença la primera finestra de tota la columna, i el màxim on acaba una finestra en la mateixa columna. Aquest mètode servirà per traçar tota la columna, incloses totes les finestres (veure la figura 29).

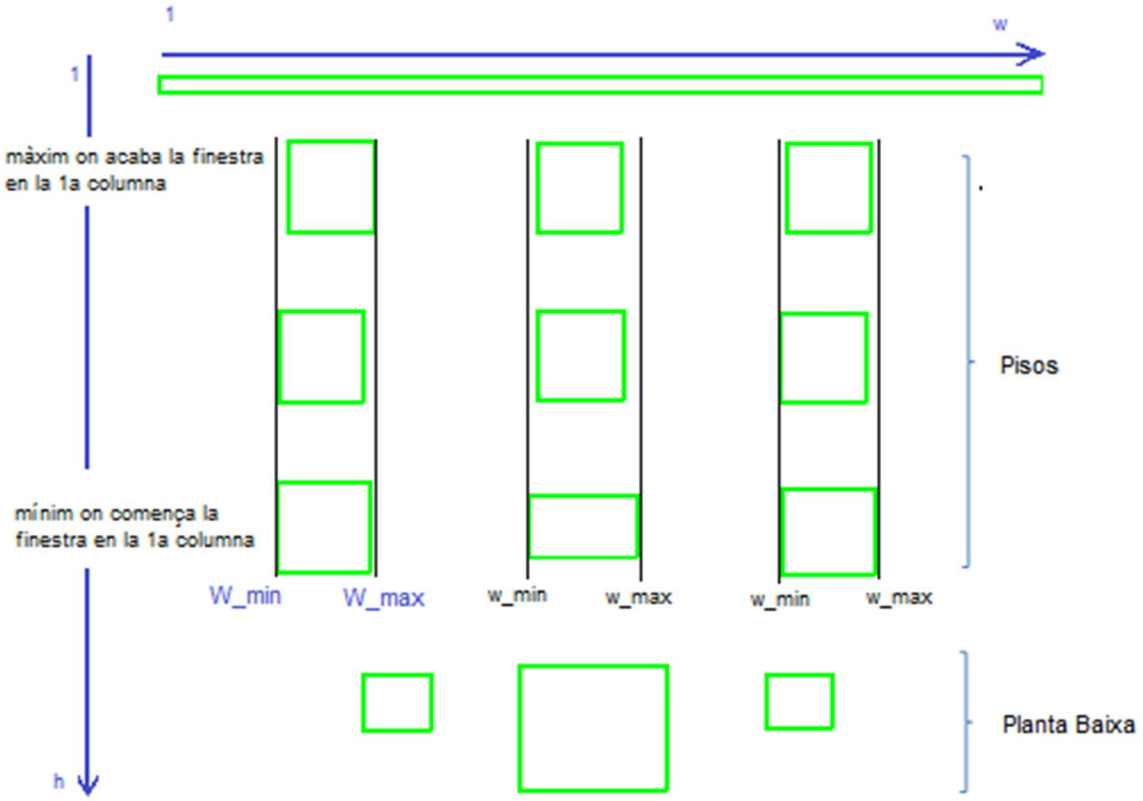


Figura 29: Assignació mínim màxim d'amplada de finestres per columna

En la figura 30 veiem, després de delimitar els pisos i les columnes, com ha quedat la façana.

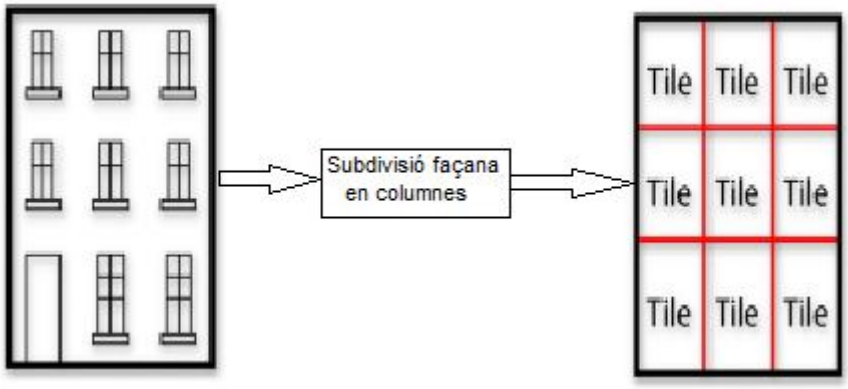


Figura 30: subdivisió façana en columnes

- **DelimitarTextura**

Funció que retorna una matriu que conté la posició on comença cada columna i on acaba la textura a partir de la taula de sortida de la funció **TrobarMinMaxFinestresPerColumna.** En les figures 31 i 32 es pot veure com l'algorisme ha trobat les posicions de les textures.

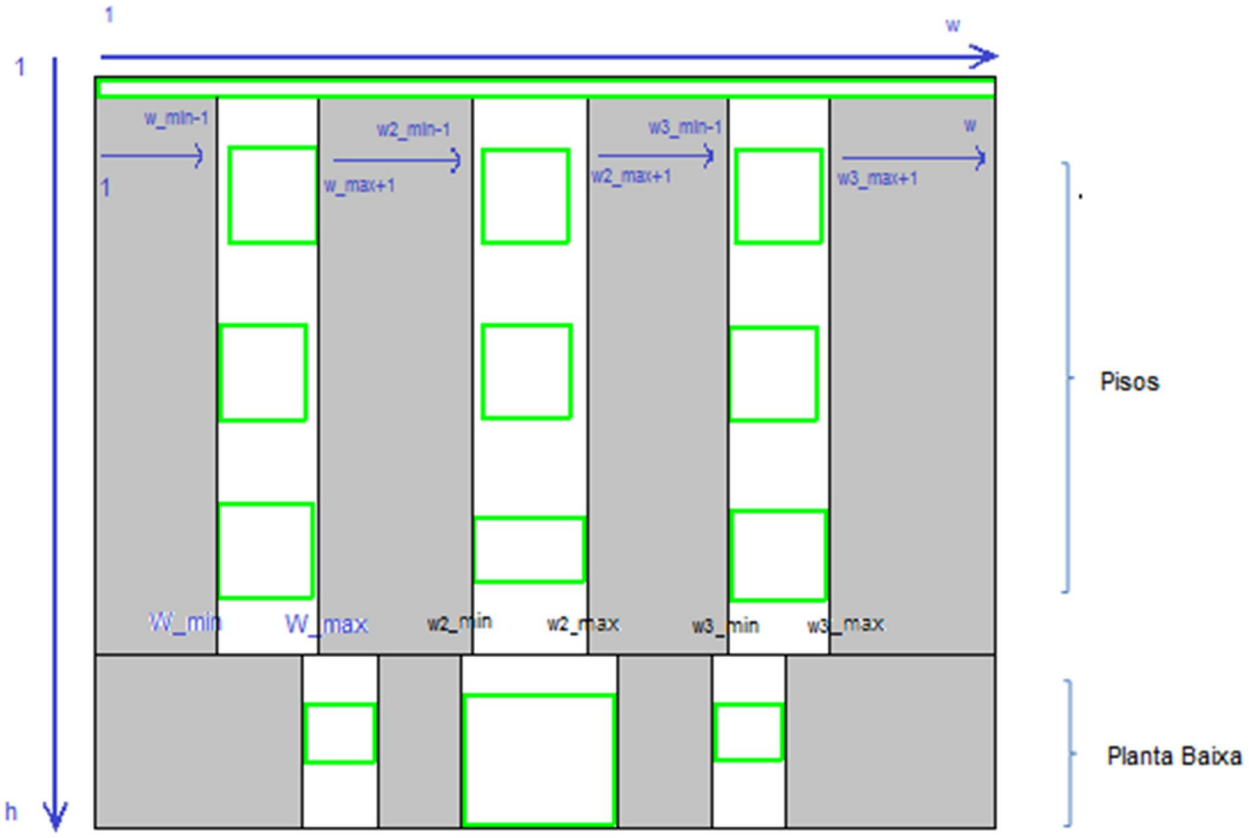


Figura 31: Textura de la façana en color gris

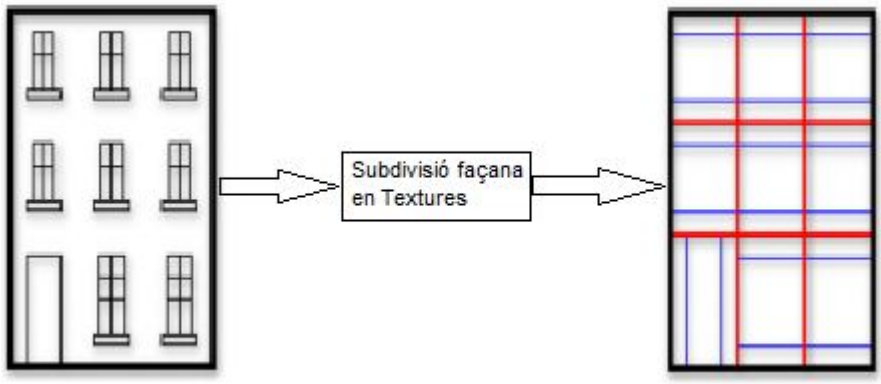


Figura 32: façana resultant de la subdivisió en pisos i columnes i textures

- **PosicioPorta**

Funció que retorna la posició de la porta a partir de la taula d'objectes trobats. Si ens fixem en la taula de la Figura 26, veiem en la posició 6 que té l'altura més gran de tots els altres. Això indica que és la porta (veure també la figura 33).

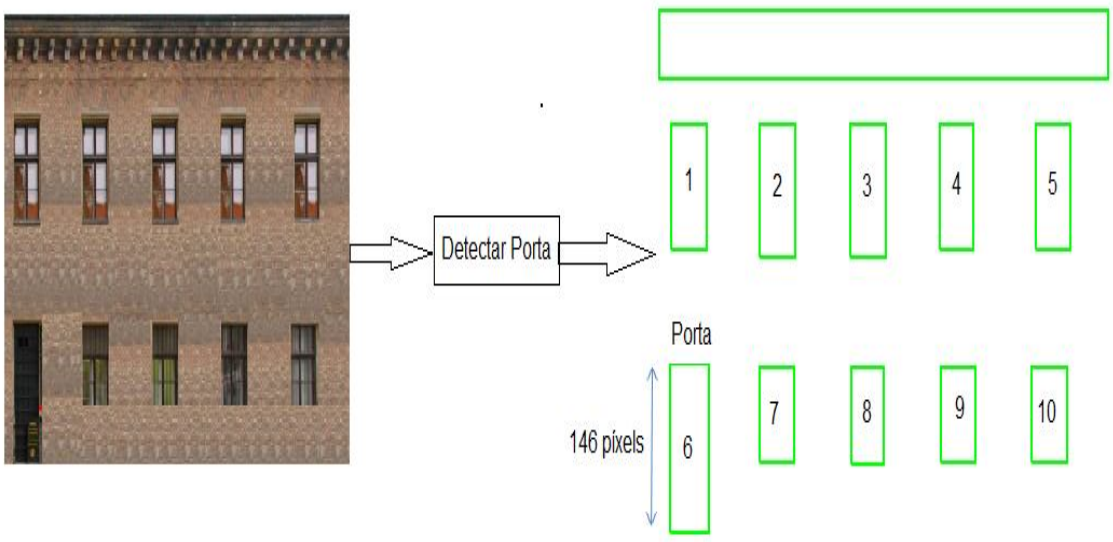


Figura 33: Posició de la porta en la façana

- **GenerarFitxerExplicitOriginal**

Funció que retorna un fitxer .txt explícit generat a partir de la imatge d'entrada.

Després dels passos anteriors ja tenim les imatges de finestres, textures i portes ben definides i guardades en format .jpg, com es mostra en la figura 34.

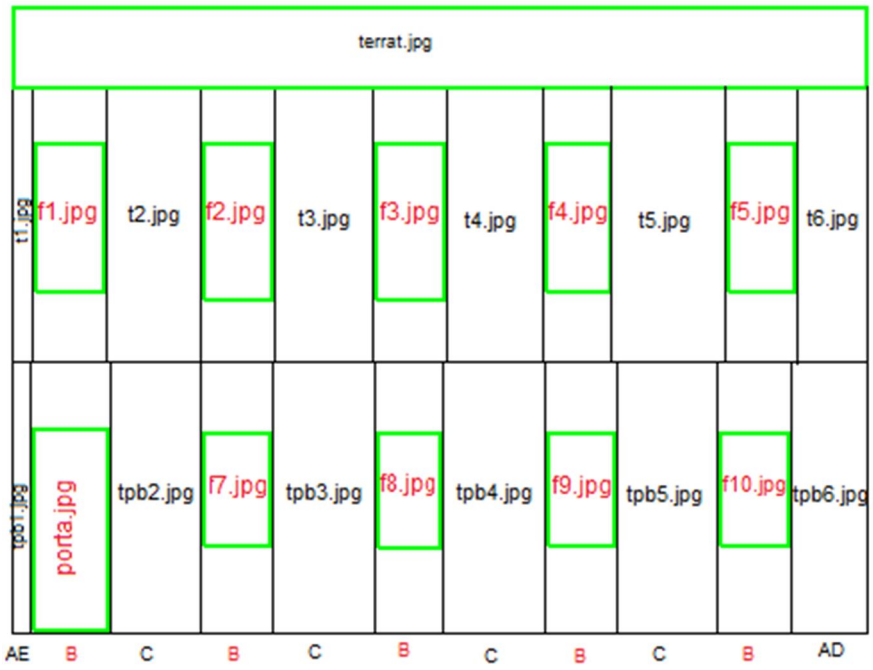


Figura 34: Componentes de la façana amb els noms dels fitxers corresponents.

A continuació podem veure la regla que s'ha fet servir per generar les imatges:

$$Fa = T\ (F)^*\ G$$

On:

- T: Terrat (Top)
- F: Pisos (Floor)
- G: Planta baixa (Ground)
- *: número de pisos que cal generar

A continuació es mostra a la figura 35 un exemple aplicant aquesta regla.

$$Fa = T\ F3\ F2\ F1\ G$$

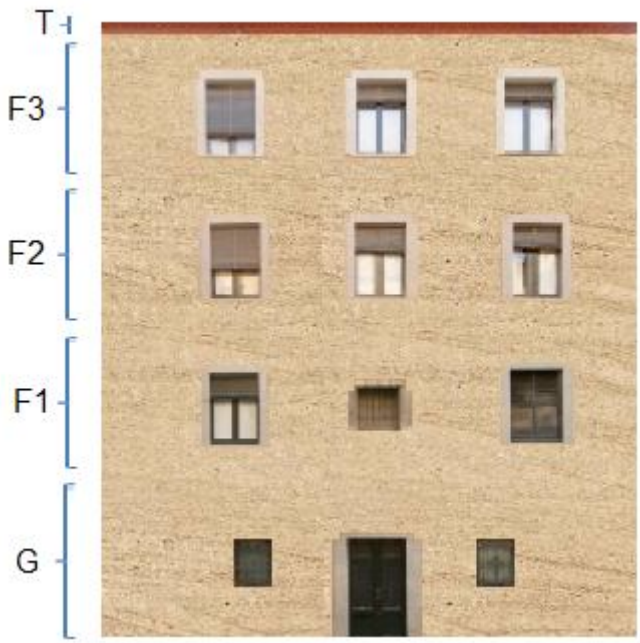


Figura 35: façana subdividida verticalment

- Regla per generar una planta F:

$$F = AE\ B\ (CB)^*\ AD$$

- AE: Textura de la part esquerre
- B: Finestra o balcó o porta
- C: Textura del mig
- AD: Textura de la part dreta de la façana
- *: Repeticions que cal generar per planta

A continuació es mostra, a la figura 36, un exemple aplicant aquesta regla.

$$F = AE\ B\ (CB)^2\ AD$$

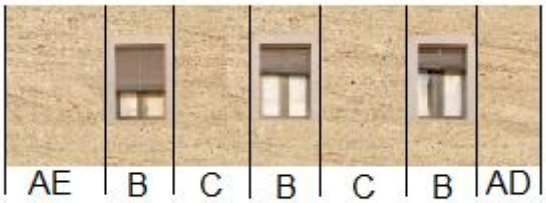


figura 36: subdivisions horitzontalment segon la regla

Aquí definim tots els components de la façana i com es generen els arxius explícit i implícit, com es mostra en la figura 34.

- fx.jpg (x és de 1 fins número de components trobats): noms d'arxius de finestres
- tx.jpg (x és de 1 fins el número de textura, que sempre és igual al número de finestres més 1): noms de les textures de les columnes.
- tpbx.jpg (x és de 1 fins el número de textures de la planta baixa) noms de les textures de la planta baixa

El contingut del fitxer explícit consta de tres parts organitzades com les etapes d'una canonada. El que fa aquesta sistema es transformar una imatge en un arxiu .txt. S'ha fet servir la subdivisió jeràrquica de dalt a baix. La primera parts es T: Terrat amb altura i amplada convertides a metres, la segona es Fi (i es el número de pisos començat per dalt fins el primer pis), la tercera és G: planta baixa. Un exemple de sortida explicita es pot veure en la figura 37 d'aquest document.

T (altura amplada terrat.jpg)
F1 AE(altura amplada t1.jpg) B(altura amplada f2.jpg) C(altura amplada t2.jpg) B(altura amplada f3.jpg) C(altura amplada t3.jpg) B(altura amplada f4.jpg) C(altura amplada t4.jpg) B(altura amplada f5.jpg) C(altura amplada t5.jpg) B(altura amplada f6.jpg) AD(altura amplada t6.jpg)
G AE(altura amplada tpb1.jpg) B(altura amplada porta.jpg) C(altura amplada tpb2.jpg) B(altura amplada f7.jpg) C(altura amplada tpb3.jpg) B(altura amplada f8.jpg) C(altura amplada tpb4.jpg) B(altura amplada f9.jpg) C(altura amplada tpb5.jpg) B(altura amplada f10.jpg) AD(altura amplada tpb6.jpg)

Figura 37: arxiu explícit d'una imatge

- **GeneralFitxerImplicitOriginal**

Funció que retorna un fitxer .txt implícit generat a partir de la imatge d'entrada, Igual que el fitxer explícit, la diferència amb respecte els pisos és que no cal recórrer tot, Només cal posar un pis i després posar asterisc i el número de pisos (veure figura 38).

T (altura amplada terrat.jpg)
F1 AE(altura amplada t1.jpg) B(altura amplada f2.jpg) C(altura amplada t2.jpg) B(altura amplada f3.jpg) C(altura amplada t3.jpg) B(altura amplada f4.jpg) C(altura amplada t4.jpg) B(altura amplada f5.jpg) C(altura amplada t5.jpg) B(altura amplada f6.jpg) AD(altura amplada t6.jpb)***1**
G AE(altura amplada tpb1.jpg) B(altura amplada porta.jpg) C(altura amplada tpb2.jpg) B(altura amplada f7.jpg) C(altura amplada tpb3.jpg) B(altura amplada f8.jpg) C(altura amplada tpb4.jpg) B(altura amplada f9.jpg) C(altura amplada tpb5.jpg) B(altura amplada f10.jpg) AD(altura amplada tpb6.jpg)

Figura 38: arxiu implícit d'una imatge

- **DibuixarTerrat**

Funció que, a partir de la altura i amplada, retorna una línia en negre que és el terrat de la façana generada. En una matriu de altura 2 píxels i de l'amplada de la imatge generada (veure figura 39).

Figura 39: imatge del terrat de la imatge generada

- **GenerarFaçana**

Funció que genera una façana a partir dels paràmetres de entrada, que són número de pisos i número de finestres per planta que cal generar (veure l'exemple d'execució de la figura 43 del capítol 9).

En la figura 40 veiem un exemple de generar una façana de 2 pisos i 3 finestres per planta a partir d'una imatge d'entrada real.

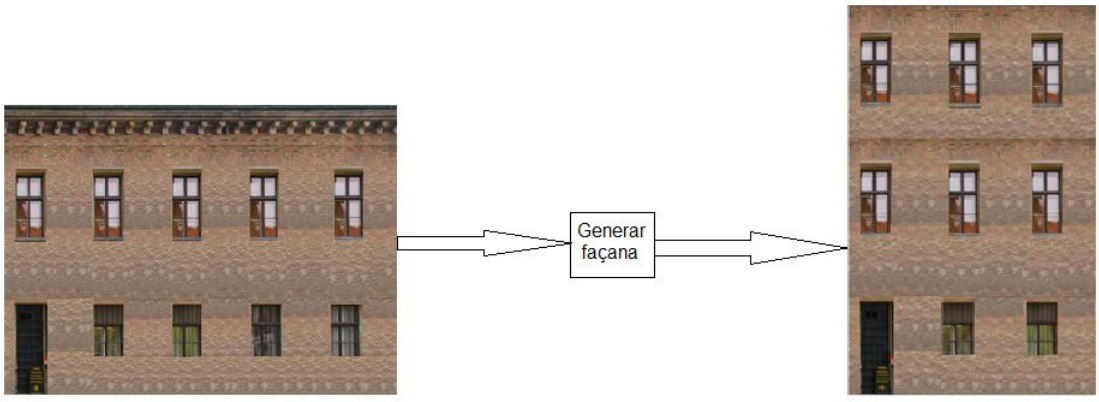


Figura 40: imatge generada amb piso = 2 i número de finestres per planta = 3

- **GenerarFaçanaFinestresExplicit**

Funció que retorna un fitxer .txt explícit generat a partir de la imatge generada, (Veure la taula de la figura 26)

- **GenerarFaçanaFinestresImplicit**

Funció que retorna un fitxer .txt implícit generat a partir de la imatge generada. (Veure la taula de la figura 26)

Capítol 9: Implementació i proves

9.1 Implementacions

Aquest capítol parla sobre els passos que s'han seguit a l'hora de realitzar la interfície gràfica i altres detalls d'implementació.

Classe Interfície

Interfície
Img: matriu n*m*3 (3: RGB)
Adjuntar() GravarSortidaExplicita() GravarSortidaImplicita() Autors() obtNumeroPisos() obtNumeroFinestresPerPis() IniciFaçana() generarPlantes() OmplirPlanta() FerPlantaBaixaFinestres() FerPlantaBaixaPortesFinestres() GravarFaçanaGenerat() GravarSortidaExplicitaGenerada() GravarSortidaImplicitaGenerada()

Tot seguit veiem pseudocodi de algunes funcions de la classe Interfície.

- **GravarSortidaExplicita**

Aquest codi guarda una descripció de la façana actual en un arxiu .txt, unint parts de textura i elements (finestres, balcons) dels diferents pisos de la imatge d'entrada.

FUNCIÓ *GravarSortidaExplicita (t_pisos, p: taula, h_inici, h_BA, w_prin, pisos, pos_porta: enter w_c, c_baix, finestres_ord: matriu de n files i 4 columnes, fid: fitxer)*
*{**PRE**::t_pisos: taula que conté el número d'elements per pis, p: taula que conté les posicions dels pisos on comencan i on acaban, h_inici: alçada on acaba el Terrat, h_BA: alçada on comença la planta baixa, w_prin: amplada d'imatge d'entrada, pisos: número de pisos, pos_porta: la posició de la porta de la façana, w_c: taula que conté les "y" on comencan les textures i on acaban, c_baix: taula que conté les "y" on comencan les textures i on acaban de la planta baixa, finestres_ord: matriu de n files i*

```
4 columnes ordanda per alçada x}
{POST:: retorna un fitxer .txt explícit generat a partir de la imatge d'entrada }

SI h_inici > 1 LLAVORS          % si la façana té terrat llavors escriure les dades del
   altura = h_inici;                % terrat al fitxer
   altura = ConvertirPixelMetres(altura, h_pis);
   amplada = getAmpladaFaçana();
   amplada = ConvertirPixelMetres(amplada, w_pis);
   fprintf(fid, "T (", altura, amplada, terrat.jpg, ")\n");
FSI
j = 1;
PER i DE 1 FINS pisos FER          % fer tots els pisos
   fprintf(fid, "F ", (pisos-i+1));  % comencem per últim pis fins la planta baixa fem la part AE,
   altura  = p[2*i] – p[2*i – 1];    % altura de la primera planta
   amplada_c = w_c[1];     %l'amplada de la part AE sempre es lo mateix en totes les
   [altura, amplada_c] = ConvertirPixelMetres(altura, amplada-c, h_pis); % plantes
   fprintf(fid, "AE(", altura, amplada_c, c1.jpg, ")"); % escriure en el fitxer explícit la part AE
   amplada = finestres_ord[i][3];    % la llargada de l'amplada
   amplada = ConvertirPixelMetres(amplada, w_pis);
   k = 1;
   MENTRE j <= suma(t(1:i)) FER
      fprintf(fid, 'B', altura, amplada, 'f', j, '.jpg ');          % la part de les finestres o balcons
      SI  j != suma (t(1:i)) LLAVORS
          amplada = w_c[k];          % taula con conté les posicions de la textura de la façana
          amplada = ConvertirPixelMetres(amplada, w_pis);
          fprintf(fid, 'C', altura, amplada, 'textura', j, '.jpg ');
      FSI
       j++;   k++;
   FMENTRE
amplada = w_c[end];
amplada = ConvertirPixelMetres(amplada, w_pis);
fprintf(fid, 'AD(', altura, amplada, 'textura_AD', '.jpg) \n');
FPER
% comença la planta baixa
Altura = h_BA;                          % altura de la planta baixa
Altura = ConvertirPixelMetres(altura, h_BA);
Amplada = c_baix[2] – c_baix[1] + 1;
Fprintf(fid, 'G');                      % escriure la lletra de la planta baixa
j = 1;

PER  i DE  suma(t_pisos[1:pisos])+1 FINS suma(t_pisos)
   Amplada = finestres_ord[i][3];
   Amplada = ConvertirPixelMetres(amplada, w_pis);
   SI i != pos_porta LLAVORS
      fprintf(fid, 'B(', altura, amplada, 'finestra', i, '.jpg ');
   ALTRAMENT
     fprintf(fid, 'B(', altura, amplada, 'porta', i, '.jpg ');
FSI
j ++;
FPER
Amplada = c_baix[2*j] – c_baix[2*j-1];    % l'amplada de la textura de la planta baixa
Amplada = ConvertirPixelMetres(amplada, h_BA);
fprintf(fid, 'C(', altura, amplada, 'tb, i, '.jpg ');
FFUNCIÓ
```

- **GravarSortidaImplicita**

El codi d'aquesta funció es semblant a GravarSortidaExplicita, amb la diferència en el primer PER, que s'ha de treure, perquè només cal recórrer un pis. Després de comença la planta baixa cal afegir aquesta línia:

fprintf(fid, '', pisos, '\n');* *% que es el número de pisos*

A continuació es mostra l'algorisme de com generar una façana a partir d'uns paràmetres d'entrada.

- **GenerarFaçana**

Aquest codi es basa de enganxar les parts de la imatge que tenim guardats en format .jpg, en una imatge nova amb alçada i amplada calculades (veure figura 41), començant pel terrat, després l'últim pis i acabant pel primer pis (veure figura 42).

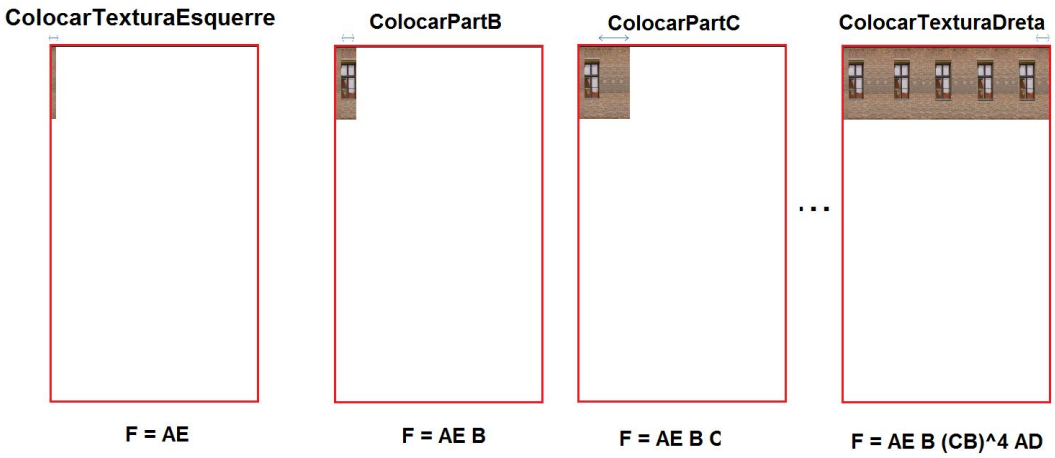


Figura 41: Generar una planta

Figura 42: Generar pisos

A continuació abans de generar una façana es calcula l'amplada i l'alçada d'aquesta nova imatge com es pot veure en la funció IniciFaçana.

```
FUNCIÓ IniciFaçana(finestres_ord, w_c, num_FinPis, textura, numPisos)
{PRE:: finestres_ord: taula que conté tots els elements ordenats per x, w_c: taula que conté les posiciones de les textures, numFinPis: el número de finestres que cal posar en una planta, numPisos: el número de pisos que cal generar, }
{POST::retorna la imatge que cal generar en blanc, i la imatge de la dimensió d'una planta amb el seu amplada}

%calcular l'amplada de la nova planta de la imatge que vol generar
[altura, amplada_planta] = CalcularAlturaAmpladaNovaImatge(finestres_ord, w_c, num_FinPis, textura, numPisos);

% comencem pel últim pis
w_AE = c[2] – c[1];              % obtenir la textura de part esquerre
w_AD = c[end] – c[end -1];   % obtenir la textura de part dreta
h1_img = 1;
h2_img = 2;

% Crear una imatge en blanc amb amplada i altura calculat
img_nova = crearImatge(altura, amplada_planta);

% Crear la imatge planta per col·locar els trossos de finestres i textures
planta = crearImatge(h_pis, amplada_planta);

RETORNA <img_nova, planta>

FFUNCIÓ
```

Després de tenir l'amplada i l'alçada de la nova imatge que cal generar, ja es pot generar plantes amb la funció generarPlantes.

```
FUNCIÓ generarPlantes(img_nova, planta, numPisos, numFinPis, p, pisos, t_pisos, w_b, w_c, h_BA, c_baix, h_pis, h_prin, finestres_ord, im, h_b, pis_mig, p)
{PRE:: planta: imatge de la dimensió d'una planta de la imatge que cal generar, , numPisos: el número de pisos que cal generar, numFinPis: el número de finestres que cal posar en una planta, p: taula que conté les posicions dels pisos, pisos: el número de pisos en la imatge d'entrada, t_pisos: taula que conté el número d'elements en cada pis, w_b: taula que conté les posiciones de les finestres o balcons, w_c: taula que conté les posiciones de les textures, h_BA: l'altura de la planta baixa, c_baix: taula que conté les posicions de les textures de la planta baixa, h_prin: l'altura de la imatge d'entrada, finestres_ord: taula que conté tots els elements ordenats per x,  im: imatge de la entrada per copiar la textura, h_b: al altura del pis del mig,  pis_mig: el pis del mig de la imatge d'entrada, p: taula que conté les posicions de les altures dels pisos}
{POST::retorna imatge generat amb pisos i finestres i amb planta baixa en blanc}

% EL PRIMER BUCLE PER LES PLANTES
% col·locar el terrat en la imatge nova per fer els pisos
img_nova = ColocarTerratImatgeNova(img_nova, amplada_planta);
```

```
PER i DE 1 FINS numPisos FER
    h_pisInici = 1;
    h_pisFi = h_b;
    w_pisInici = 1;
    w_pisFi = w_AE;
   % Fer la part AE, sempre es lo mateix en totes les plantes.
    planta = ColocarTexturaEsquerre(im, pis_mig, h_pisInici, h_pisFi, w_pisInici,
w_pisFi);

% EL SEGON BUCLE PER OMPLIR LES PLANTES
    planta = OmplirPlanta(planta, w_pisFi, numFinPis, pisos, t_pisos, w_b, w_c,
finestres_ord);
% Fer la part AD
w1 = w_c[end-1];
w2 = w1 + (amplada_planta – w_pisFi) -1;
img_AD = transformarImatge(im, p[pis_mig], p[pis_mig+1], w1, w2);
w_pisInici = w_pisFi + 1;
w_pisFi = amplada_planta;
Planta = ColocarTexturaDreta(c, amplada_planta, w_pisFi, im, p, pis_mig, planta,
h1_img, h2_img);
% Col·locar la planta construïda a la imatge nova fins la planta baixa
h1_img = h2_img + 1;
h2_img = h_pisFi*i+2;
img_nova = CopiarSubImatge(img_nova, planta, h1_img, h2_img, 1,amplada_planta);
FPER
% FINS AQUI TENIM EL TERRAT I ELS PISOS FETS, FALTA FER LA PLANTA
BAIXA
img_nova = FerPlantaBaixaFinestres(img_nova);

RETORNA <img_nova>
FFUNCIÓ
```

A continuació la funció OmplirPlanta esta cridat per la funció generarPlantes, el que fa es omplir la planta per finestres i textures de forma aleatòria.

```
FUNCIÓ OmplirPlanta (planta, w_pisFi, numFinPis, pisos, t_pisos, w_b, w_c,
finestres_ord)
{PRE:: planta: imatge de la dimensió d'una planta de la imatge que cal generar,
w_pisFi: la posició on acaba l'amplada, numFinPis: el número de finestres que cal
posar en una planta, pisos: el número de pisos en la imatge d'entrada, t_pisos: taula
que conté el número d'elements en cada pis, w_b: taula que conté les posiciones de
les finestres o balcons, w_c: taula que conté les posiciones de les textures,
finestres_ord: taula que conté tots els elements ordenats per x }
{POST:: retorna la imatge d'una planta construïda amb finestres i textures }

PER j DE 1 FINS numFinPis FER
    % FER LA PART B QUE ÉS FINESTRA O BALCÓ
    rand_b[k] = suma(t_pisos[1:pisos]);
    [h1, h2, w1, w2] = PosarDades(finestres_ord, rand_b[k]);
    w2 = w1 + w2;
    w_pisInici = w_pisFi + 1;
    w_pisFi = w_pisFi + w_b;
```

```
    planta = ColocarPartB(t_pisos, finestres_ord, rand_b, h_pisInici, h_pisFi,
                                    w_pisInici, w_pisFi, planta, k);

  % FER LA PART C LA TEXTURA
  SI j != numFinPis LLAVORS
       w_pisInici = w_pisFi + 1;
       mida = w_c[rand_textura[j]];
       w_pisFi = w_pisFi + mida;
       w1 = c[2*rand_textura[j]-1];
       w2 = c[2*rand_textura[j]];
      planta = ColocarPartC(numFinPis, w_pisFi, w_c, rand_textura, j, planta, h1, h2);
  FSI

   k++;
  FPER

RETORNA <planta>
FFUNCIÓ
```

- **FerPlantaBaixaFinestres**

Un cop fet el terrat i tots els pisos que cal generar, només falta la planta baixa, que cal fer amb dos opcions: el primer es fer-la amb portes i finestres, o amb repeticions de només finestres. Aquest és el cas de la funció següent.

```
FUNCIÓ FerPlantaBaixaFinestres(img_nova, h_BA, c_baix, h_prin, p,
amplada_planta, t_pisos, pisos, finestres_ord)
{PRE:: img_nova: imatge generat amb planta baixa en blanc, h_BA: l'altura de la
planta baixa, c_baixa: taula que conté les amplades de la textura de la planta baixa,
h_prin: l'altura de la imatge d'entrada, p: taula que conté les altures dels pisos,
amplada_planta es la amplada nova de la imatge que cal generar }
{POST::retorna imatge final amb planta baixa generat}

%la planta baixa
hb_fi = size(img_nova);
hb_inici = hb_fi – h_BA + 1;
ant = 1;
pertany = true;
seg = 0;
volta = 1;
MENTRE  pertany  FER
    %la part de la textura esquerre
    w1 = 1;
    w2 = c_baix[2];
    SI volta == 1 LLAVORS
         seg = seg + c_baix(2);
    ALTRAMENT
        seg = ant + w2 – 1;
    FSI
    img_nova = ColocarSubImatge(img_nova, im, p, h_prin, w1, w2, ant, seg,
    hb_inici, hb_fi, amplada_planta);
```

```
    w1 = w2 + 1;
    ant = seg + 1;
    j = suma(t_pisos(1:pisos))+1;
    k = 1;  l = 2;
    SI !ok || volta = 1  LLAVORS
        Img_nova = OmplirPlantaBaixa(img_nova, h_BA, c_baix, h_prin, p,
amplada_planta, t_pisos, pisos, finestres_ord);
    FSI
    MENTRE   ok  && seg < amplada_planta    FER
        % fer la part B
        w1 = finestres_ord(j)(1);
        w2 = finestres_ord(j)(1) + finestres_ord(j)(3) + 1;
        seg = seg + finestres_ord(j)(3) + 2;
        img_nova = ColocarSubImatge(img_nova, im, p, h_prin, w1, w2, ant, seg,
        hb_inici, hb_fi, amplada_planta);
        % fer la part de la textura
        w1 = w2 + 1;
        ant = seg + 1;
        k = k + 1;
        w1 = c_baix(2*k-1);
        w2 = c_baix(2*k);
        seg = seg + (w2 –w1)+1;
        SI  seg < amplada_planta   LLAVORS
            Img_nova = ColocarSubImatge(img_nova, im, p, h_prin, w1, w2, ant,
            seg, hb_inici, hb_fi, amplada_planta);
            w1 = w2 + 1;
            ant =seg + 1;
        ALTRAMENT
            pertany = false;
        FSI
        j++, l++;
    FMENTRE
    SI  seg > amplada_planta LLAVORS
        pertany = false;
    FSI
      volta++;
FMENTRE

FFUNCIÓ
```

A continuació es mostra el mètode OmplirPlantaBaixa(). Simplemente consisteix en omplir la planta baixa amb textures, finestres i portes.

```
FUNCIÓ OmplirPlantaBaixa(img_nova, h_BA, c_baix, h_prin, p, amplada_planta,
t_pisos, pisos, finestres_ord)
{PRE:: img_nova: imatge generat amb planta baixa en blanc, h_BA: l'altura de la
planta baixa, c_baixa: taula que conté les amplades de la textura de la planta baixa,
h_prin: l'altura de la imatge d'entrada, p: taula que conté les altures dels pisos,
amplada_planta es la amplada nova de la imatge que cal generar }
{POST:: retorna la imatge de la planta baixa construïda amb finestres, portes i
textures }

MENTRE   j <= suma(t_pisos)  &&  seg < amplada_planta  FER
            % fer la part B si hi ha espai suficient
```

```
            w2 = finestres_ord(j)(1) + finestres_ord(j)(3)+1;
            SI volta == 1 LLAVORS
                seg = finestres_ord(j)(1) + finestres_ord(j)(3)+1;
            ALTRAMENT
                seg = seg + finestres_ord(j)(3) + 2;
            FSI
            img_nova = ColocarSubImatge(img_nova, im, p, h_prin, w1, w2, ant, seg,
            hb_inici, hb_fi, amplada_planta);
            %escriure la part B
            w1 = w2 + 1;
            ant = seg +1;
            % fer la part de la textura
            k++;
            w2 = c_baix(2*k);
            SI  volta == 1  LLAVORS
                seg = c_baix(2*k);
            ALTRAMENT
                seg = seg + (w2 –w1)+1;
            FSI
            SI  seg < amplada_planta LLAVORS
                img_nova =ColocarSubImatge(img_nova, im, p, h_prin, w1, w2, ant,
                seg, hb_inici, hb_fi, amplada_planta);
                w1 = w2 + 1;
                ant = seg + 1;
            ALTRAMENT
                pertany = false;
            FSI
            j++;  l++;
        FMENTRE
RETORNA <img_nova>
FFUNCIÓ
```

- **CalcularAlturaAmpladaNovaImatge**

El codi següent es mostra l'altura i la amplada calculades a partir de la nova imatge que cal generar.

```
FUNCIÓ CalcularAlturaAmpladaNovaImatge(finestres_ord, w_c, num_FinPis,
numPisos)
{PRE::finestres_ord: taula que conté tots els elements ordenats per x, w_c: taula que
conté les posicions de la textura, num_FinPis: el número de finestres que cal posar en
una planta, numPisos: el número de pisos que cal generar}
{POST::Retorna altura, amplada de la imatge que cal generar}
%calcular l'amplada de la nova planta de la imatge que vol generar
amplada_b = finestres_ord(1)(3);
amplada_c = w_c(1) + w_c(end);
PER i DE 1 FINS numFinPis FER
    rand_c(i) = random(size(textura));
    amplada = amplada_c + w_c(rand_textura(i));
FPER
% la llargada de l'amplada de la primera finestra multiplicat per el número de finestres per
```

```
planta sumant l'amplada de la textura c calculat
amplada_planta = numFinPis*amplada_b + amplada_c;

% l'altura de la imatge nova es l'altura del pis del mig multiplicat pel número de pisos que cal generar més l'altura de la planta baixa.
altura = h_b*numPisos + h_BA;

RETORNA <altura, amplada>

FFUNCIÓ
```

- **ColocarTerratImatgeNova**

Aquest codi és el començament de la fase generarFaçana. El que fa és enganxar la part del terrat, que és una línia en color negre, en la imatge que cal generar (veure figura 41).

```
FUNCIÓ ColocarTerratImatgeNova (img_nova, amplada_planta)
{PRE:: img_nova: imatge en blanc, amplada_planta es la amplada nova de la imatge que cal generar}
{POST::retorna una imatge en blanc amb una línia recta en color negre a dalt}

h_pisInici = 1;
h_pisFi = 2;
%enganxar el terrat en la imatge nova
img_T = DibuixarTerrat(h_pisInici, h_pisFi, amplada_planta);
img_nova = CopiarSubImatge(img_t, img_nova, h_pisInici, h_pisFi, 1, amplada_planta);

RETORNA <img_nova>
FFUNCIÓ
```

- **ColocarTexturaEsquerre**

Aquest codi el que fa es enganxar la part Esquerre de la textura de la imatge d'entrada en una posició donada en la imatge que cal generar (veure figura 41).

```
FUNCIÓ ColocarTexturaEsquerre(planta, im, pis_mig, h_pisInici, h_pisFi, w_pisInici, w_pisFi)
{PRE::planta: la imatge que cal colocar la textura, im: imatge de la entrada per copiar la textura, pis_mig: el pis del mig de la imatge d'entrada, h_pisInici: altura on comença a col·locar la textura, h_pisFi: altura on acaba, w_pisInici, la amplada on comença a col·locar la textura, w_pisFi: la amplada on acaba}
{POST:: retorna una imatge amb textura Esquerre col·locat en la posició donat}

% Fer la part AE, sempre es lo mateix en totes les plantes.
   Img_AE = transformarImatge(im, p(pis_mig), p(pis_mig)+h_pisFi, w_pisInici,
```

```
   w_pisFi);
   planta = CopiarSubImatge(planta, img_AE,1, h_pisFi-h_pisInici+1, 1, w_pisFi-
   w_pisInici+1);

RETORNA  <planta>
FFUNCIÓ
```

- **ColocarPartB**

Igual que ColocarPartC però ara enganxar la part B (finestra, balcó) de la imatge d'entrada en una posició donat en la imatge que cal generar (veure figura 41).

```
FUNCIÓ  ColocarPartB(t_pisos, finestres_ord, rand_b, h_pisInici, h_pisFi, w_pisInici,
w_pisFi, planta, k)
{PRE::t_pisos: taula que conté número de components en cada pis, finestres_ord:
taula que conté totes les components ordenats per x, rand_b: taula que conté
números aleatoris de les finestres que cal colocar, h_pisInici: altura on comença a
colocar l'element, h_pisFi: altura on acaba, w_pisInici: amplada on comença a colocar
l'element, w_pisFi: amplada on acaba}
{POST:: retorna una imatge amb Finestra o balcó col·locat en la posició donat }

img_b = transformarImatge(im, h1, h2, w1, w2);
planta = CopiarSubImatge( planta, img_b, h_pisInici, h_pisFi, w_pisInici,
                                  w_pisFi);

RETORNA  <planta>
FFUNCIÓ
```

- **ColocarPartC**

Aquest codi el que fa es enganxar la part de la textura de la imatge d'entrada en una posició donat en la imatge que cal generar (veure figura 41).

```
FUNCIÓ ColocarPartC(numFinPis, w_pisFi, w_c, rand_textura, j, planta, h1, h2)
{PRE:: planta: la imatge que cal colocar la textura, im: imatge de la entrada per copiar
la textura, h1: altura on comença a col·locar la textura, h2: altura on acaba, w_pisFi: la
amplada on acaba, w_c: taula que conté les posicions de la textura, rand_textura:
taula que conté números aleatoris de la textura que cal colocar, numFinPis: el número
de pis que cal generar}
{POST:: retorna una imatge amb textura col·locat en la posició donat}

 textura = transformarImatge(im, h1, h2, w1, w2);
planta = CopiarSubImatge(planta, textura, h_pisInici, h_pisFi, w_pisInici,
                                  w_pisFi);
```

```
RETORNA   <planta>
FFUNCIÓ
```

- **ColocarTexturaDreta**

Amb aquest codi que enganxar la part de la textura dreta de la imatge d'entrada en una posició donat en la imatge que cal generar terminem la planta (veure figura 41).

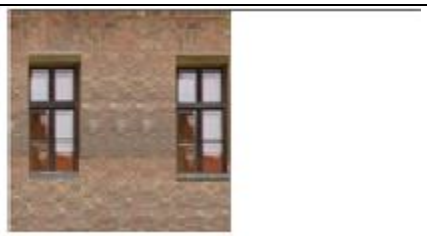
```
FUNCIÓ ColocarTexturaDreta(planta, im, pis_mig, h_pisInici, h_pisFi, w_pisInici,
w_pisFi)
{PRE:: planta: la imatge que cal colocar la textura, im: imatge de la entrada per copiar
la textura, pis_mig: el pis del mig de la imatge d'entrada, h_pisInici: altura on comença
a col·locar la textura, h_pisFi: altura on acaba, w_pisInici, la amplada on comença a
col·locar la textura, w_pisFi: la amplada on acaba}
{POST:: retorna una imatge amb textura Dreta col·locat en la posició donat }

planta = CopiarSubImatge(planta, img_AD, h_pisInici, h_pisFi, w_pisInici, w_pisFi);
img_nova = CopiarSubImatge(img_nova, planta, h1_img, h2_img, 1, amplada_planta);

RETORNA  < planta>
FFUNCIÓ
```

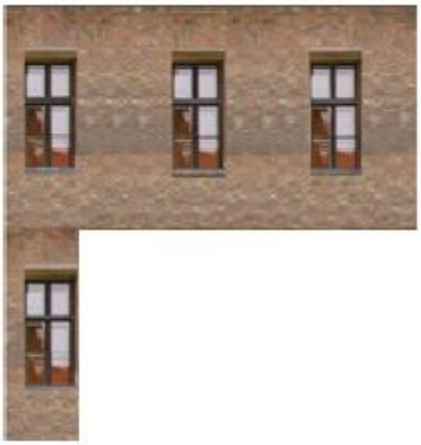
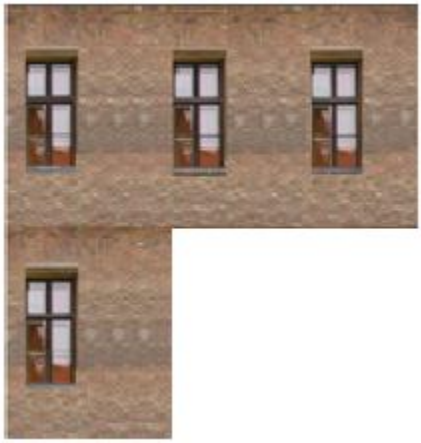
9.2 Exemple de execució

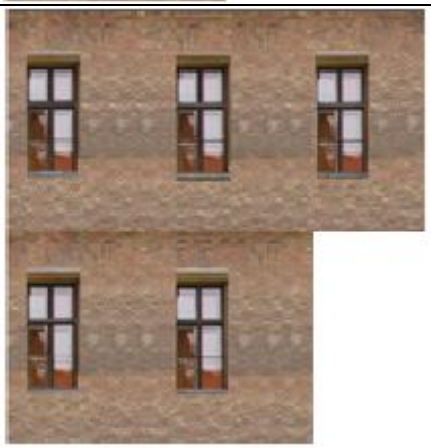
A continuació presentarem un exemple mostrant els passos detallats de la fase de generar façana a partir de l'exemple de la figura 34: (amb "fx.jpg" com a nom de finestres, amb x = 1....número de finestres):

	Primerament s'ha de començar per dalt, això s'ha d'enganxar el terrat i en **el fitxer explícit** esta escrit : T(w h terrat.jpg) **Fitxer implícit:** T(w h terrat.jpg)
	Enganxar la textura de la part esquerra (AE). **Fitxer explícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) **Fitxer implícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg)
	Enganxar la finestra escollida aleatòriament. **Fitxer explícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) **Fitxer implícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg)
	Enganxar la textura escollida aleatòriament. **Fitxer explícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) **Fitxer implícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg)
	Enganxar la finestra escollida aleatòriament. **Fitxer explícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) **Fitxer implícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg)

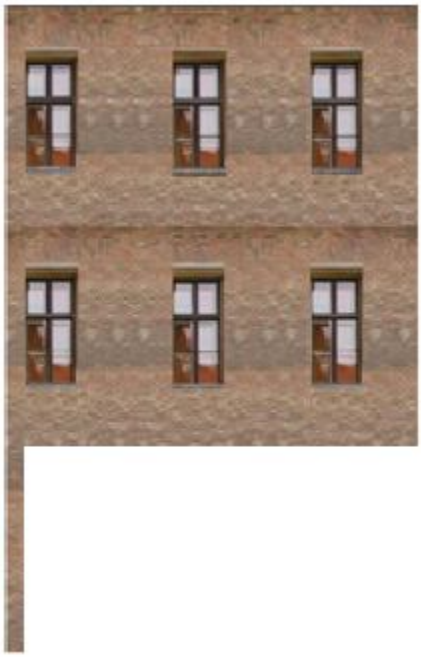
	Enganxar la textura escollida aleatòriament. **Fitxer explícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) **Fitxer implícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg)
	Enganxar la finestra escollida aleatòriament. **Fitxer explícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) **Fitxer implícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg)
	Enganxar la textura de la part dreta (AD) **Fitxer explícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg) **Fitxer implícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg)

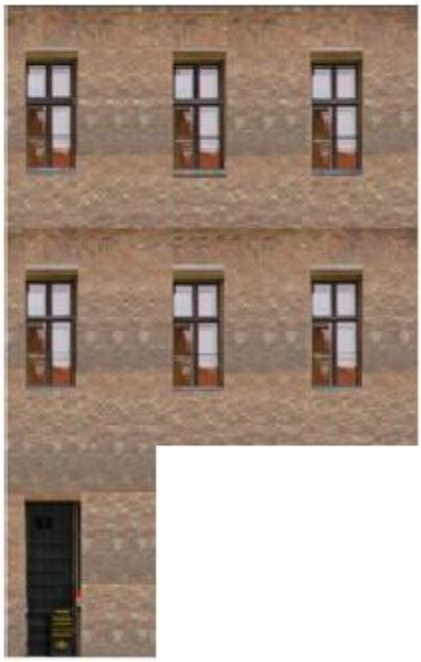
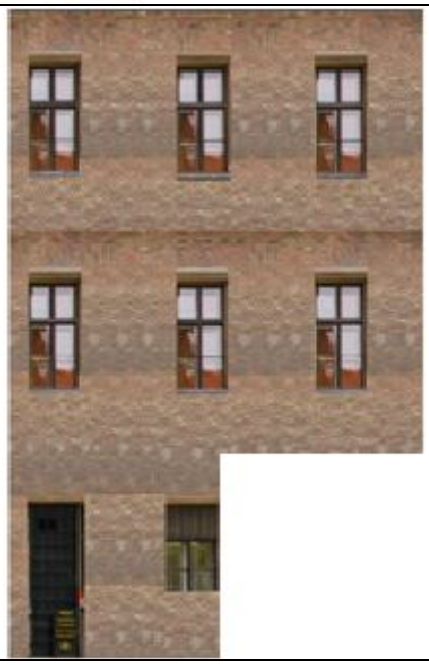
Fins aquí hem fet el terrat i el segon pis. A continuació fem el mateix amb el primer pis.

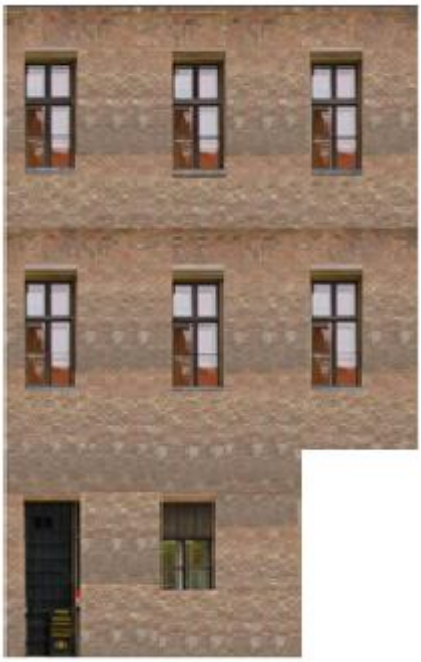
	Enganxar la textura d'esquerra AE. **Fitxer explícit:** T(w h terrat.jpg) F2 **AE(w1 h1 t1.jpg)** B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg) F1 AE(w1 h1 t1.jpg) **Fitxer implícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg) F1 AE(w1 h1 t1.jpg)
	Enganxar la finestra aleatòriament **Fitxer explícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg) F1 AE(w1 h1 t1.jpg) B(w8 h8 fx.jpg) **Fitxer implícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg)***2**
	Enganxar la textura **Fitxer explícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) **C(w3 h3 tx.jpg)** B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg) F1 AE(w1 h1 t1.jpg) B(w8 h8 fx.jpg) C(w3 h3 tx.jpg) **Fitxer implícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg)***2**

	Enganxar la finestra aleatòriament **Fitxer explícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg) F1 AE(w1 h1 t1.jpg) B(w8 h8 fx.jpg) C(w3 h3 tx.jpg) B(w9 h9 fx.jpg) **Fitxer implícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg)***2**
	Enganxar la textura **Fitxer explícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) **C(w5 h5 tx.jpg)** B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg) F1 AE(w1 h1 t1.jpg) B(w8 h8 fx.jpg) C(w3 h3 tx.jpg) B(w9 h9 fx.jpg) C(w5 h5 tx.jpg) **Fitxer implícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg)***2**
	Enganxar la finestra aleatòriament **Fitxer explícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg) F1 AE(w1 h1 t1.jpg) B(w8 h8 fx.jpg) C(w3 h3 tx.jpg) B(w9 h9 fx.jpg) C(w5 h5 tx.jpg) B(w10 h10 fx.jpg) **Fitxer implícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg)***2**
	Enganxar la finestra aleatòriament **Fitxer explícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) **AD(w7 h7 t6.jpg)** F1 AE(w1 h1 t1.jpg) B(w8 h8 fx.jpg) C(w3 h3 tx.jpg) B(w9 h9 fx.jpg) C(w5 h5 tx.jpg) B(w10 h10 fx.jpg) AD(w7 h7 t6.jpg) **Fitxer implícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg)***2**

El següent pas es enganxar els components de la planta baixa:

	Enganxar la textura AE de la planta baixa **Fitxer explícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg) F1 AE(w1 h1 t1.jpg) B(w8 h8 fx.jpg) C(w3 h3 tx.jpg) B(w9 h9 fx.jpg) C(w5 h5 tx.jpg) B(w10 h10 fx.jpg) AD(w7 h7 t6.jpg) G AE(w11 h11 tx.jpg) **Fitxer implícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg)***2** G AE(w11 h11 tx.jpg)
	Enganxar la porta **Fitxer explícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg) F1 AE(w1 h1 t1.jpg) B(w8 h8 fx.jpg) C(w3 h3 tx.jpg) B(w9 h9 fx.jpg) C(w5 h5 tx.jpg) B(w10 h10 fx.jpg) AD(w7 h7 t6.jpg) G AE(w11 h11 tpb1.jpg) B(w12 h12 porta.jpg) **Fitxer implícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg)***2** G AE(w11 h11 tpb1.jpg) B(w12 h12 porta.jpg)

	Enganxar la textura **Fitxer explícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg) F1 AE(w1 h1 t1.jpg) B(w8 h8 fx.jpg) C(w3 h3 tx.jpg) B(w9 h9 fx.jpg) C(w5 h5 tx.jpg) B(w10 h10 fx.jpg) AD(w7 h7 t6.jpg) G AE(w11 h11 tpb1.jpg) B(w12 h12 porta.jpg) C(w13 h13 tpb2.jpg) **Fitxer implícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg)***2** G AE(w11 h11 tpb1.jpg) B(w12 h12 porta.jpg) C(w13 h13 tpb2.jpg)
	Enganxar la finestra **Fitxer explícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg) F1 AE(w1 h1 t1.jpg) B(w8 h8 fx.jpg) C(w3 h3 tx.jpg) B(w9 h9 fx.jpg) C(w5 h5 tx.jpg) B(w10 h10 fx.jpg) AD(w7 h7 t6.jpg) G AE(w11 h11 tpb1.jpg) B(w12 h12 porta.jpg) C(w13 h13 tpb2.jpg) B(w14 h14 fx.jpg) **Fitxer implícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg)***2** G AE(w11 h11 tpb1.jpg) B(w12 h12 porta.jpg) C(w13 h13 tpb2.jpg) B(w14 h14 fx.jpg)

	Enganxar la textura **Fitxer explícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg) F1 AE(w1 h1 t1.jpg) B(w8 h8 fx.jpg) C(w3 h3 tx.jpg) B(w9 h9 fx.jpg) C(w5 h5 tx.jpg) B(w10 h10 fx.jpg) AD(w7 h7 t6.jpg) G AE(w11 h11 tpb1.jpg) B(w12 h12 porta.jpg) C(w13 h13 tpb2.jpg) B(w14 h14 fx.jpg) C(w15 h15 tpb3.jpg) **Fitxer implícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg)***2** G AE(w11 h11 tpb1.jpg) B(w12 h12 porta.jpg) C(w13 h13 tpb2.jpg) B(w14 h14 fx.jpg) C(w15 h15 tpb3.jpg)
	Enganxar la finestra **Fitxer explícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg) F1 AE(w1 h1 t1.jpg) B(w8 h8 fx.jpg) C(w3 h3 tx.jpg) B(w9 h9 fx.jpg) C(w5 h5 tx.jpg) B(w10 h10 fx.jpg) AD(w7 h7 t6.jpg) G AE(w11 h11 tpb1.jpg) B(w12 h12 porta.jpg) C(w13 h13 tpb2.jpg) B(w14 h14 fx.jpg) C(w15 h15 tpb3.jpg) B(w16 h16 fx.jpg) **Fitxer implícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg)***2** G AE(w11 h11 tpb1.jpg) B(w12 h12 porta.jpg) C(w13 h13 tpb2.jpg) B(w14 h14 fx.jpg) C(w15 h15 tpb3.jpg) B(w16 h16 fx.jpg)

	Enganxar la textura **Fitxer explícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg) F1 AE(w1 h1 t1.jpg) B(w8 h8 fx.jpg) C(w3 h3 tx.jpg) B(w9 h9 fx.jpg) C(w5 h5 tx.jpg) B(w10 h10 fx.jpg) AD(w7 h7 t6.jpg) G AE(w11 h11 tpb1.jpg) B(w12 h12 porta.jpg) C(w13 h13 tpb2.jpg) B(w14 h14 fx.jpg) C(w15 h15 tpb3.jpg) B(w16 h16 fx.jpg) AD(w17 h17 tpb6.jpg) **Fitxer implícit:** T(w h terrat.jpg) F2 AE(w1 h1 t1.jpg) B(w2 h2 fx.jpg) C(w3 h3 tx.jpg) B(w4 h4 fx.jpg) C(w5 h5 tx.jpg) B(w6 h6 fx.jpg) AD(w7 h7 t6.jpg)***2** G AE(w11 h11 tpb1.jpg) B(w12 h12 porta.jpg) C(w13 h13 tpb2.jpg) B(w14 h14 fx.jpg) C(w15 h15 tpb3.jpg) B(w16 h16 fx.jpg) AD(w17 h17 tpb6.jpg)

Figura 43: passos per generar una façana

9.3 Problemes trobats durant la realització del projecte

La dificultat principal del projecte ha resultat estar en l'aprenentatge de les diferents eines: Python, wxPthon, Matlab. Però tot hi això, ens hem trobat amb un seguit de punts en els quals caldrà fer especial esment. La resolució d'aquests problemes ha obligat a investigar molt i conèixer a fons el funcionament de les llibreries utilitzades.

Exemples d'aquestes problemes amb solucions

- Els diferents components trobats poden tenir mides diferents. Això es un problema perquè, a l'hora de generar la segona planta, l'amplada serà diferent que la primera que hem generat. Això s'ha solucionat al normalitzar totes els components, com es mostra en la figura 44:

Figura 44: finestres amb amplada normalitzada

Es a dir, agafar la finestra petita amb un tros de textura de la dreta i l'esquerra. Així serà fàcil col·locar-les de forma ràpida i amb amplada igual que les anteriors i les següents.

- Un altre problema és verticalment. Les finestres estan guardades com es mostra la figura 45:

Figura 45: finestres detectades

Durant la generació no s'ha pogut omplir la textura per dalt i baix de les finestres perquè no eren en la mateixa línia, i a l'hora de generar fitxer explícit i/o implícit es complica més entendre'ls (veure figura 46).

Figura 46: generar façana sense textura en les finestres

La solució va ser guardar totes les finestres amb la mateixa altura, que es l'altura del pis afegint la textura de dalt i de baix, com es mostra la figura 47.

Figura 47: finestres guardades

Capítol 10: Resultats

En aquest capítol es mostraran els resultats obtinguts seguint la metodologia descrita a l'apartat 3. Aquests resultats es poden veure en algunes captures de pantalla que acompanyen les explicacions.

10.1 Captures de pantalla de l'aplicació

A la figura 48 podem veure la interfície d'usuari. Com es pot veure, és senzilla i entenedora.

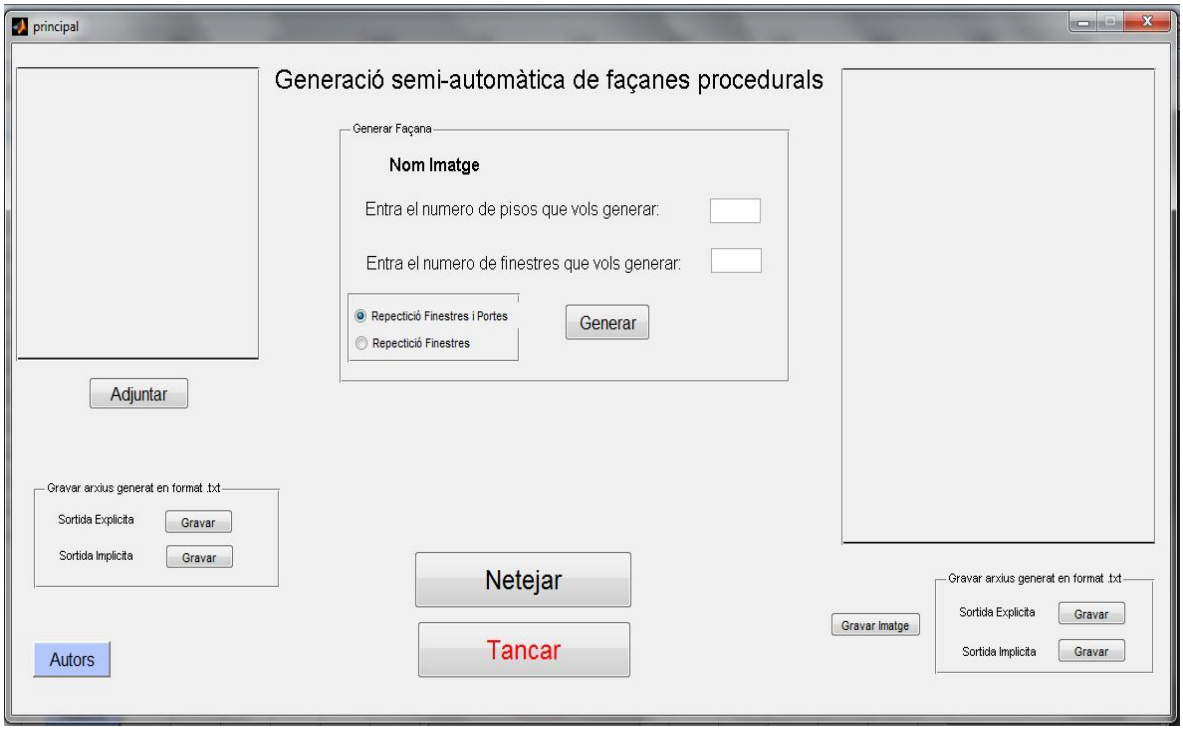


Figura 48: vista inicial de la interfície d'usuari

A la figura 49 veiem la imatge mostrat quan l'usuari ha premut el botó Adjuntar.

Figura 49: imatge mostra després de prémer el boto Adjuntar

Si l'usuari vol Gravar la sortida d'un arxiu Implícit o explícit, l'aplicació preguntarà on vol guardar l'arxiu, com es mostra a la figura 50:

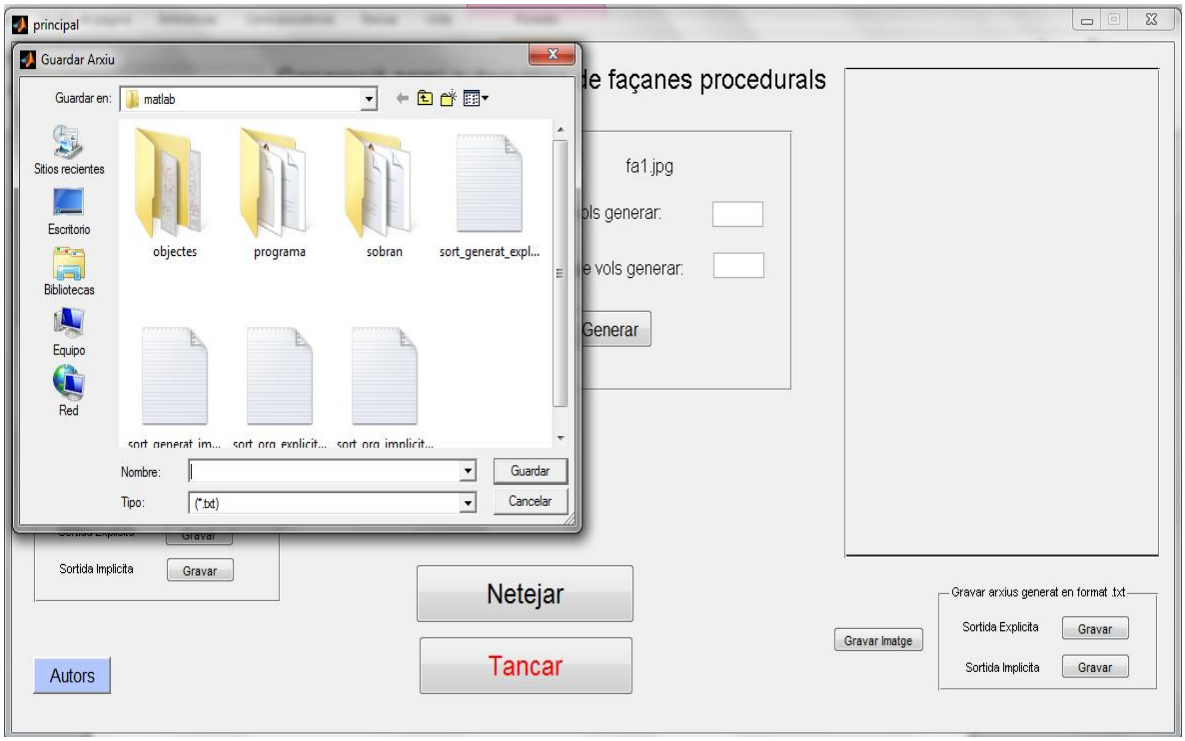


Figura 50: després de prémer el boto Gravar sortida, explicita i/o implícita

Una vegada l'usuari ha omplert els paràmetres, i ha escollit l'opció "Repetició finestres i portes" (figura 51) o "Repetició finestres" (figura 52), i prem el botó Generar, l'aplicació mostra la imatge generada (figures 51 i 52).

Figura 51: façana generat amb l'opció Repetició finestres i portes

Figura 52: façana generat amb l'opció Repetició finestres

Al prémer el boto Autors es mostrarà la imatge que es pot veure a la figura 53.

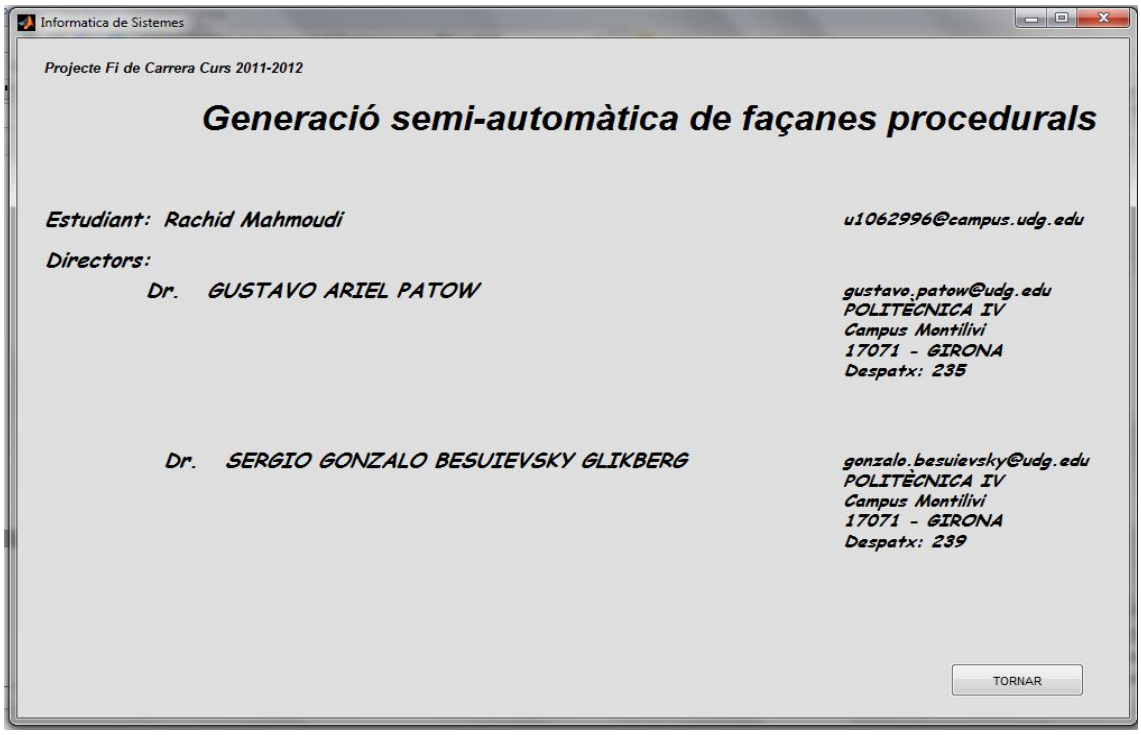


Figura 53: Títol, Autor, Directors del Projecte Fi de Carrera

A continuació es mostra uns exemples de façanes generades a les figures 54, 55 i 56:

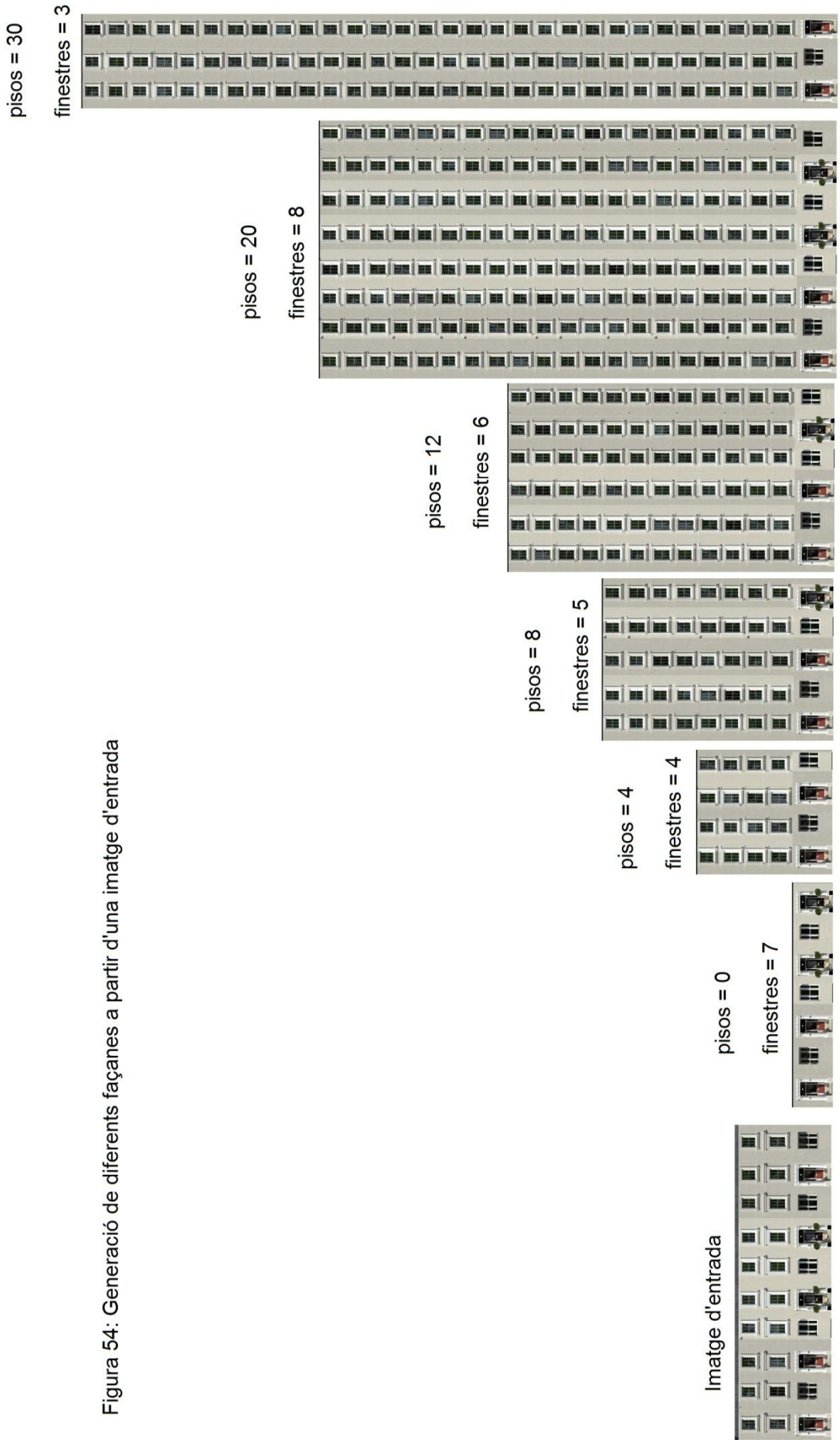


Figura 54: Generació de diferents façanes a partir d'una imatge d'entrada

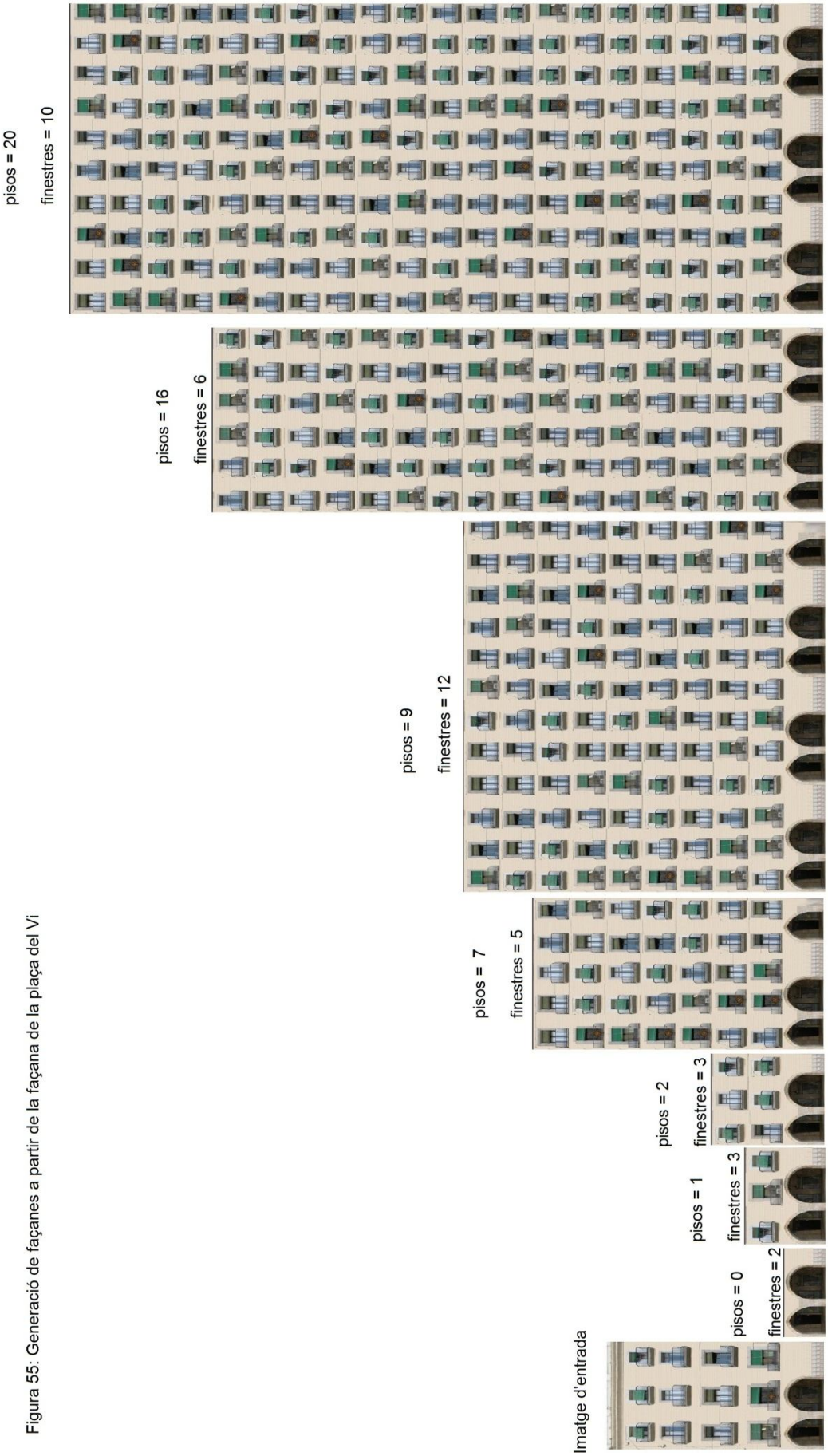


Figura 55: Generació de façanes a partir de la façana de la plaça del Vi

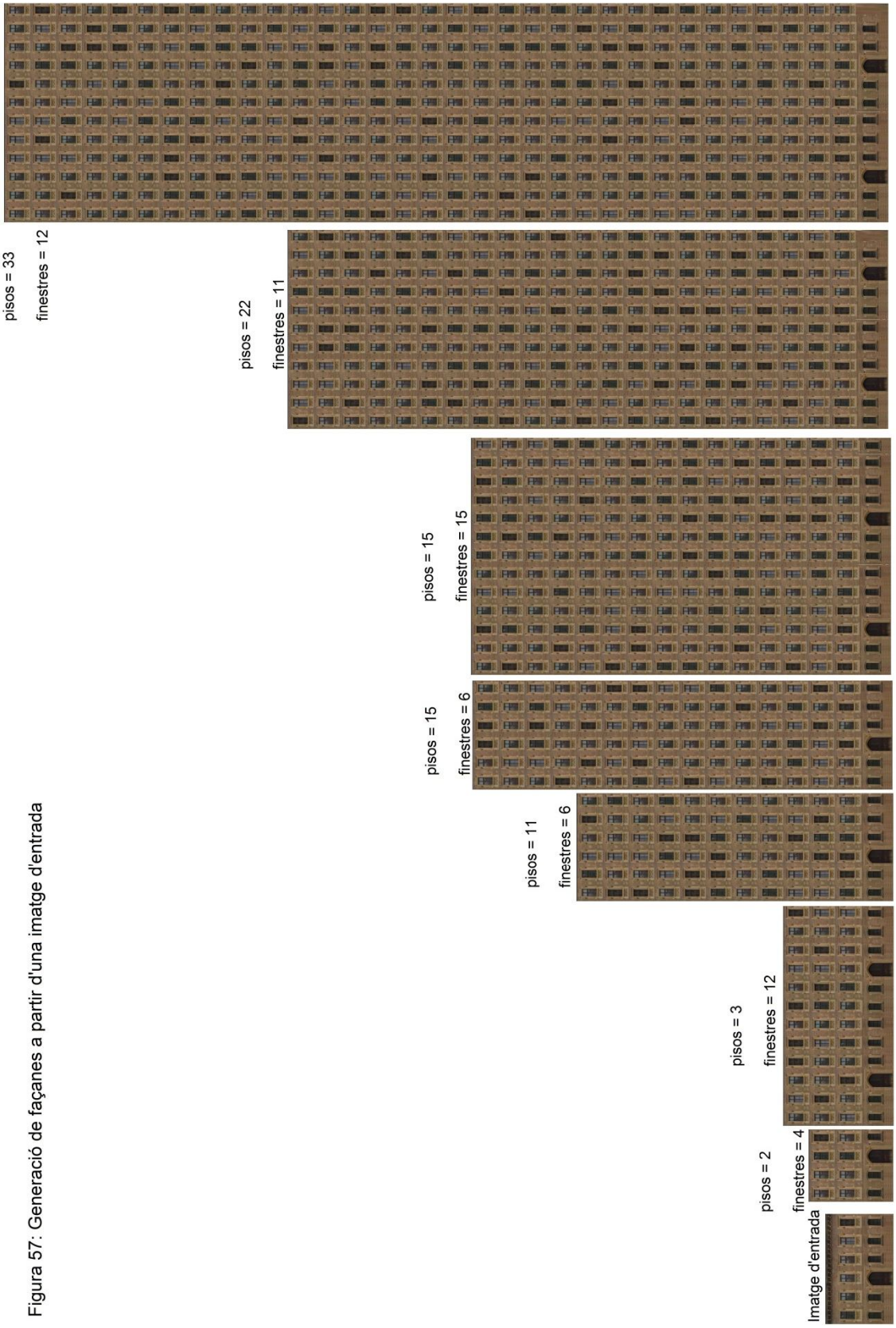


Figura 57: Generació de façanes a partir d'una imatge d'entrada

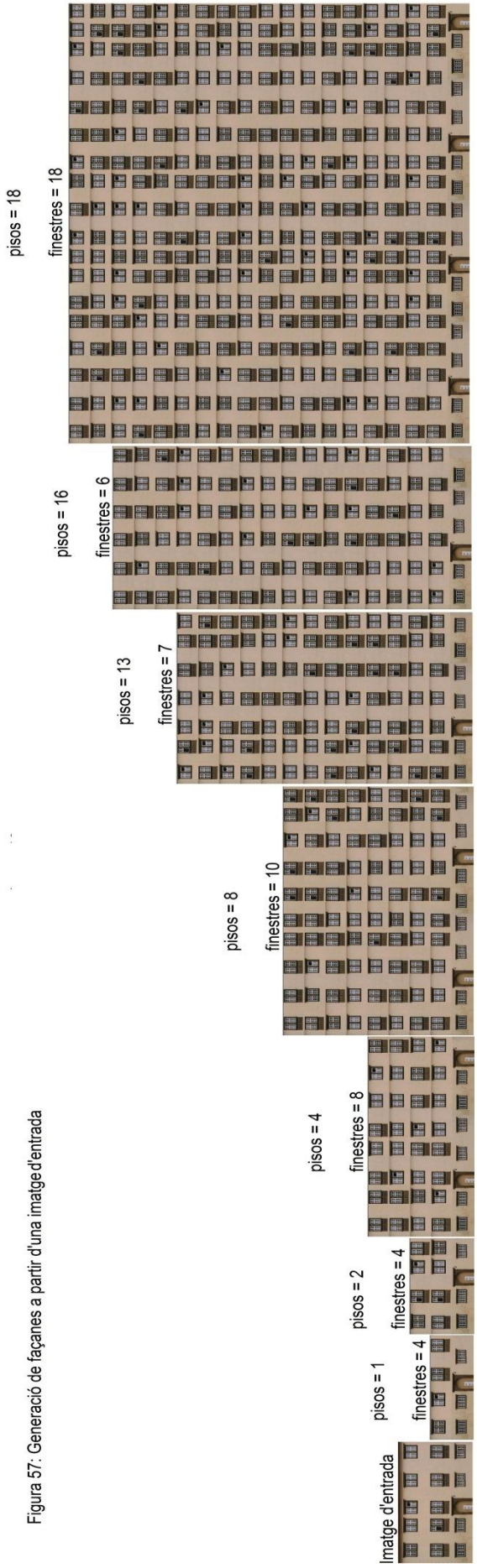


Figura 57: Generació de façanes a partir d'una imatge d'entrada

També es pot observar les diferents façanes generades amb els mateixos paràmetres. El que veiem en la figura 58 són façanes generades amb els mateixos paràmetres de número de pisos (en aquest cas 2) i el número de finestres per planta, que és 4. Però cal observar que la diferència entre una façana i altra està en les diferents finestres.

Figura 58: generació de façanes amb paràmetres iguals

Amb respecte a l'ús dels fitxers explícit i implícit, és mostraran dos façanes generades en 3D per l'aplicació **skyline**Engine incorporant aquests fitxers (veure figures 58b i 58c).

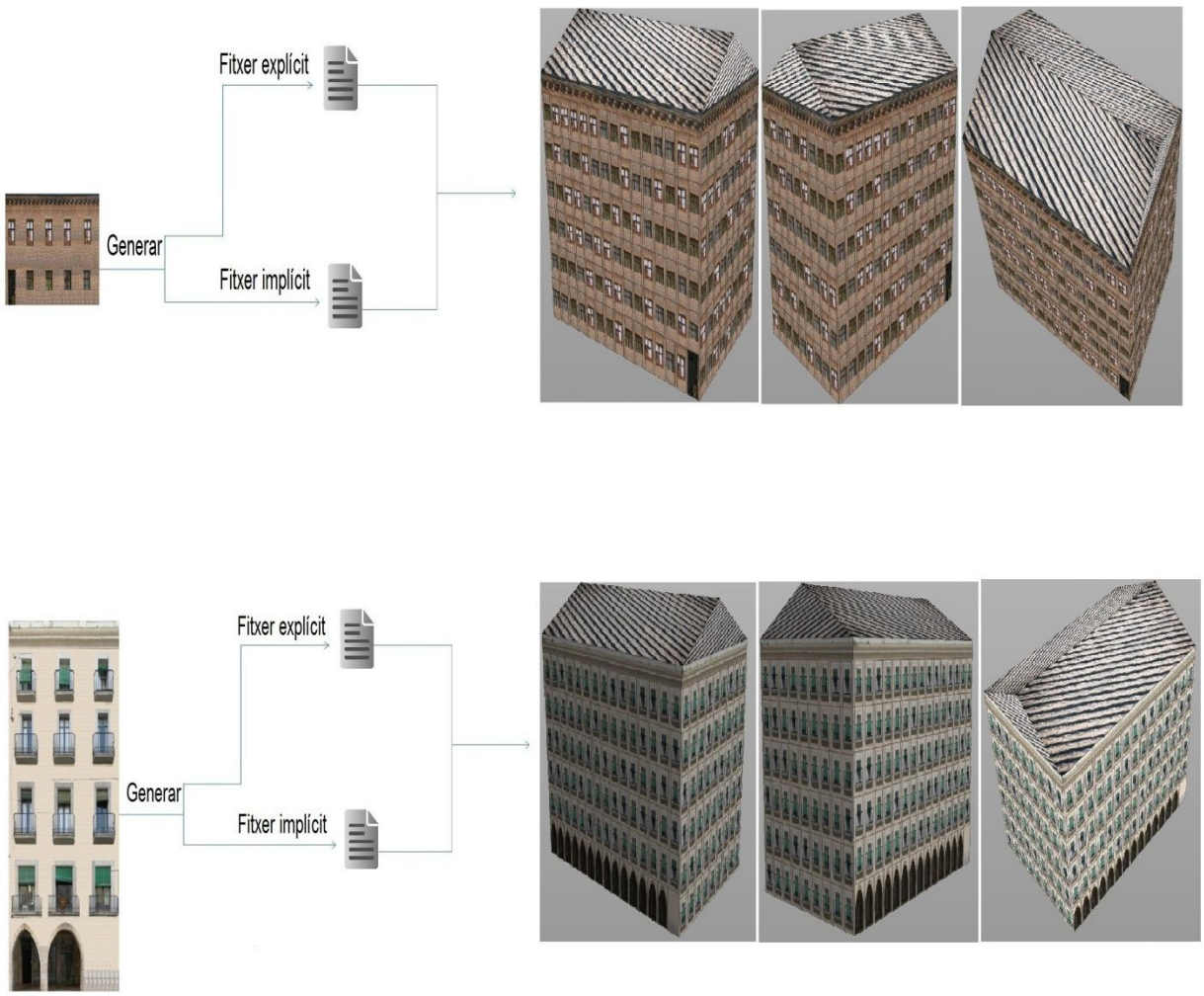


*Figura 58b: Façanes generades en 3D per **skyline**Engine utilitzant els fitxers generats per una façana d'entrada*

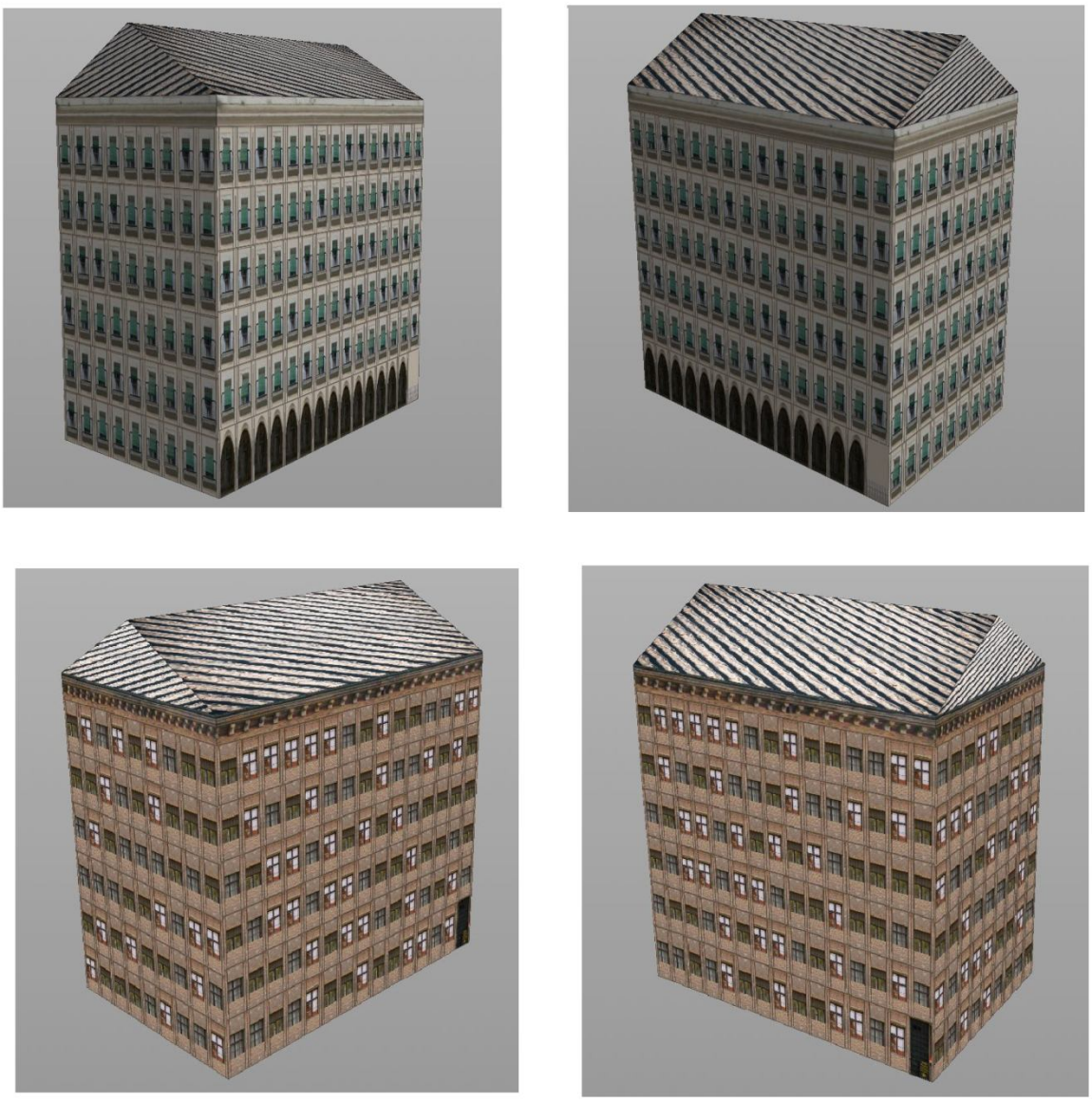


*Figura 58b: Façanes generades en 3D per **skyline**Engine utilitzant els fitxers generats per una façana d'entrada*

Capítol 11: Conclusió

L'objectiu principal d'aquest projecte consistia en crear un programari que permetés generar façanes de manera automàtica a partir d'una imatge real. Aquest objectiu ha estat complert de forma satisfactòria, així com tot el conjunt de requeriments definits des d'un principi.

En un principi, el camí a seguir era llegir els articles sobre la detecció de les finestres en una façana. Amb aquests articles s'han après les tècniques per dividir la façana per pisos i per columnes.

A més, també s'ha aconseguit desenvolupar una sèrie de llibreries per tal de instanciar interactivament els elements de la façana.

Durant la elaboració d'aquest projecte s'han arribat a nombroses conclusions, de les quals destaco:

- És fonamental utilitzar les eines adequades i realitzar un bon anàlisi dels problemes que poden sorgir. Encara que estigui més temps per arribar a una millor solució, es nota en el resultat final i s'arriben a solucionar molts de problemes.
- Realitzar un projecte gràfic no es tan senzill com començar a programar de cop sense pensar. Requereix molta dedicació i paciència.
- S'ha aprés a programar en Matlab. És un llenguatge fàcil d'aprendre, molt potent, ràpid d'implementació i que fa que el codi quedi bastant net.

Capítol 12: Treball Futur

La segmentació basada en detecció de contorns presenta diversos problemes. El més important potser consisteix en la no aparició d'un contorn o frontera que sí que existeix en la imatge real. A més, juntament amb els contorns, sol aparèixer soroll que es deriva de la pròpia naturalesa de la imatge. Això fa que pugui aparèixer un contorn fantasma que no existeixi en la realitat.

En general, el procés de la segmentació sol resultar complex degut, d'una banda, que no es té una informació adequada dels objectes a extreure i, per l'altra, al fet que en l'escena a segmentar apareix normalment soroll. És per això que l'ús de coneixement sobre el tipus d'imatge a segmentar o alguna altra informació d'alt nivell pot resultar molt útil per aconseguir la segmentació de la imatge.

Els diferents objectes que apareixen en una imatge poden ser localitzats atenent a aspectes com: els contorns o la textura. Podem diferenciar entre 3 grups de tècniques per les que calen més estudis:

- tècniques basades en umbralització.
- tècniques basades en detecció dels contorns dels objectes.
- tècniques basades en propietats locals de les regions.

A més que el treball realitzat en aquest projecte permet un cert nombre de millores i ampliacions. Tant es podrien millorar les funcionalitats del projecte com implementar-ne de noves per façanes més complicades amb diferents finestres i patrons de repeticions més complicats. Automatitzar la integració de la façana en un generador d'edificis 3D.

Capítol 13: Bibliografia

13.1 Articles llegits

- Helmut Mayer, Sergiy Reznik. “Building Façade Interpretation from Image Sequences”. ISPRS Journal of Photogrammetry and Remote Sensing Vol. XXXVI, (2005). 5 de gener 2012
- Pascal Müller, Zeng Gang, Wonka Peter, Van Gool Luc. “Image-based Procedural Modeling of Facades”. Ítem: Revista *acm siggraph* 25 (2007): 85-93. 8 de gener 2012
- Jianxiong Xaio, Tian Fang, Ping Tan, Peng Zhao, Eyal Ofek, Long Quan. “Image-based Façade Modeling”. Acm SIGGRAPH 27 (2008). 9 de gener 2012
- Philip David, “Detection of Builiding Facades in Urban Environments”. SPIE Digital Library 6978 (2008). 9 de gener 2012
- Suhib Alsisan, Niloy J. Mitra. “Variation-Factored Enconding of Facade Images”. The Eurographics Association (2012). 9 de gener 2012
- Jorge Hernández, Beatriz Marcotegui. “Morphological Segmentation of Building Façade Images”. Mines ParisTech CMM- Centre de morphologie mathématique (2009). 15 de gener 2012
- Ruisheng Wang, jeff Bach, Frank P. Ferrie. “Window Detection from Mobile LiDAR Data”, NAVTEQ Corporation (2005). 15 de gener 2012
- Viraj Kulkarni, Rohan Nagesh, Hong Wu. “Window Detection in Frontal Facades”. Department of Computer Science University of California, Berkeley (2011). 18 de gener
- Przemyslaw Musialski, Michael Wimmer, Peter Wonka. “Interactive Coherence-Based Façade Modeling”. The Eurographics Association (2012). 18 de gener 2012
- Martin Dobias. Structural Recognition of Facades. Prague: Departament of Software and Computer Science Education, 2010. < http://cmp.felk.cvut.cz/~dobiama5/thesis-final.pdf > 20 de gener 2012

13.3 llibres consultats

- Raúl González Duque. *Python para todos*. España. 2008. 5 de febrer 2012

- Andrés Marzal, Gracia Isabel. *Introducción a la programación con Python*. Departamento de Lenguajes y Sistemas Informáticos Universitat Jaume I. 2008. 8 de febrer 2012

- Noel Rappin, Dunn Robin, *wxPython in Action*. Greenwich. Manning Publications Co. 2006. 17 de febrer 2012
- Fredrik Lundh, Ellis Matthew. *Python Imaging Library Overview.* PIL 1.1.3 | March 12, 2002. 4 de març 2012

- The MathWorks, Inc. *Image Processing Toolbox 6*. Natick. The MathWorks, Inc. 2008. 18 de març

- Diego Orlando Barragán Guerrero. *Manual De Interfaz Gráfica de usuario en Matlab*. 2007. 22 de juny 2012

13.3 Enllaços web consultats

http://www.ipb.uni-bonn.de/uploads/tx_ikgpublication/wenzel07_detection_and_description_of_repeated_structures_PFG-07-07.pdf. 5 de gener 2012

http://staff.not.iac.es/~rcardenes/hg/diveintopython3-es/index.html. 25 de febrer 2012

http://www.programando.org/aprende-a-programar/primera-parte/primeros-pasos/comparaciones-y-expresiones-logicas.html. 28 de febrer 2012

http://pserv.udg.edu/Portal/Uploads/4103862/Tema4-Segmentacio.pdf. Visió per Computador Tema 4, Etis 2009-2010 *Lladó, Xavier. 2 de març 2012*

http://es.scribd.com/doc/89104653/Vision-Artificial-Con-Matlab. 23 de març 2012

http://www.gts.tsc.uvigo.es/pi/Analisis de imagenes.pdf. 5 de març 2012

http://dmi.uib.es/~ygonzalez/VI/Material_del_Curso/Teoria/Tema5_Filtrado.pdf. 7 de març 2012

http://en.wikipedia.org/wiki/Gaussian_filter. 7 de març 2012

http://en.wikipedia.org/wiki/K-means_algorithm. 7 de març 2012

http://www.exa.unicen.edu.ar/catedras/pdi/FILES/TE/CP1.pdf. 8 de març 2012

http://www.mathworks.com/support/functions/. A partir del 15 de març fins el juliol 2012

Capítol 14: Annexos

Annex 1: (Fitxer explícit de la figura 6)

```
T(0.28m, 7.7m,  t_original.jpg)

F3 AE(2.5m, 1.4m,  c1.jpg) B(2.5m, 0.73m,  f1.jpg) C(2.5m, 1.4m,
c1.jpg) B(2.5m, 1.4m,  f2.jpg) C(2.5m, 1.2m,  c2.jpg) B(2.5m, 1.2m,
f3.jpg) AD(2.5m, 1m,  c4.jpg)

F2 AE(2.4m, 1.4m,  c1.jpg) B(2.4m, 0.73m,  f4.jpg) C(2.4m, 1.4m,
c4.jpg) B(2.4m, 1.4m,  f5.jpg) C(2.4m, 1.2m,  c5.jpg) B(2.4m, 1.2m,
f6.jpg) AD(2.4m, 1m,  c4.jpg)

F1 AE(2.5m, 1.4m,  c1.jpg) B(2.5m, 0.75m,  f7.jpg) C(2.5m, 1.4m,
c7.jpg) B(2.5m, 1.4m,  f8.jpg) C(2.5m, 1.2m,  c8.jpg) B(2.5m, 1.2m,
f9.jpg) AD(2.5m, 1m,  c4.jpg)

G AE(2.5m, 1.4m,  cb1.jpg) B(2.5m, 0.56m,  f10.jpg) C(2.5m, 0.81m,
cb2.jpg) B(2.5m, 1.2m,  p11.jpg) C(2.5m, 0.72m,  cb3.jpg) B(2.5m,
0.58m,  f12.jpg) C(2.5m, 1.6m,  cb4.jpg)
```

Annex 2 (Fitxer implícit de la figura 6)

```
T(0.28m, 7.7m,  t_original.jpg)

F AE(2.5m, 1.4m,  c1.jpg) B(2.5m, 0.73m,  f1.jpg) C(2.5m, 67m,
c1.jpg) B(2.5m, 67m,  f2.jpg) C(2.5m, 57m,  c2.jpg) B(2.5m, 57m,
f3.jpg) AD(2.5m, 1m,  c4.jpg)*3

G AE(2.5m, 1.4m,  cb1.jpg) B(2.5m, 0.56m,  f10.jpg) C(2.5m, 0.81m,
cb2.jpg) B(2.5m, 1.2m,  p11.jpg) C(2.5m, 0.72m,  cb3.jpg) B(2.5m,
0.58m,  f12.jpg) C(2.5m, 1.6m,  cb4.jpg)
```

Annex 3: Fitxer explícit de la figura 6 d'imatge generada

```
T(0.043m, 24m,  t.jpg)
F10 AE(0.043m, 1.4m,  c1.jpg) B(0.043m, 0.75m,  f9.jpg) C(0.043m,
1.4m,  c3.jpg) B(0.043m, 0.75m,  f1.jpg) C(0.043m, 1.2m,  c2.jpg)
B(0.043m, 0.75m,  f9.jpg) C(0.043m, 1.4m,  c3.jpg) B(0.043m, 0.75m,
f2.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m, 0.75m,  f7.jpg) C(0.043m,
1.4m,  c3.jpg) B(0.043m, 0.75m,  f6.jpg) C(0.043m, 1.4m,  c3.jpg)
B(0.043m, 0.75m,  f7.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m, 0.75m,
f4.jpg) C(0.043m, 1.4m,  c3.jpg) B(0.043m, 0.75m,  f6.jpg) C(0.043m,
1.2m,  c2.jpg) B(0.043m, 0.75m,  f8.jpg) C(0.043m, 1.2m,  c2.jpg)
B(0.043m, 0.75m,  f1.jpg) AD(0.043m, 1m,  c12.jpg)
```

```
F9 AE(0.043m, 1.4m,  c1.jpg) B(0.043m, 0.75m,  f1.jpg) C(0.043m, 1.4m,
c3.jpg) B(0.043m, 0.75m,  f9.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m,
0.75m,  f6.jpg) C(0.043m, 1.4m,  c3.jpg) B(0.043m, 0.75m,  f3.jpg)
C(0.043m, 1.2m,  c2.jpg) B(0.043m, 0.75m,  f4.jpg) C(0.043m, 1.4m,
c3.jpg) B(0.043m, 0.75m,  f2.jpg) C(0.043m, 1.4m,  c3.jpg) B(0.043m,
0.75m,  f3.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m, 0.75m,  f3.jpg)
C(0.043m, 1.4m,  c3.jpg) B(0.043m, 0.75m,  f3.jpg) C(0.043m, 1.2m,
c2.jpg) B(0.043m, 0.75m,  f2.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m,
0.75m,  f4.jpg) AD(0.043m, 1m,  c12.jpg)
F8 AE(0.043m, 1.4m,  c1.jpg) B(0.043m, 0.75m,  f2.jpg) C(0.043m, 1.4m,
c3.jpg) B(0.043m, 0.75m,  f8.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m,
0.75m,  f1.jpg) C(0.043m, 1.4m,  c3.jpg) B(0.043m, 0.75m,  f9.jpg)
C(0.043m, 1.2m,  c2.jpg) B(0.043m, 0.75m,  f4.jpg) C(0.043m, 1.4m,
c3.jpg) B(0.043m, 0.75m,  f1.jpg) C(0.043m, 1.4m,  c3.jpg) B(0.043m,
0.75m,  f4.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m, 0.75m,  f7.jpg)
C(0.043m, 1.4m,  c3.jpg) B(0.043m, 0.75m,  f8.jpg) C(0.043m, 1.2m,
c2.jpg) B(0.043m, 0.75m,  f5.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m,
0.75m,  f7.jpg) AD(0.043m, 1m,  c12.jpg)
F7 AE(0.043m, 1.4m,  c1.jpg) B(0.043m, 0.75m,  f9.jpg) C(0.043m, 1.4m,
c3.jpg) B(0.043m, 0.75m,  f1.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m,
0.75m,  f3.jpg) C(0.043m, 1.4m,  c3.jpg) B(0.043m, 0.75m,  f1.jpg)
C(0.043m, 1.2m,  c2.jpg) B(0.043m, 0.75m,  f2.jpg) C(0.043m, 1.4m,
c3.jpg) B(0.043m, 0.75m,  f7.jpg) C(0.043m, 1.4m,  c3.jpg) B(0.043m,
0.75m,  f7.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m, 0.75m,  f8.jpg)
C(0.043m, 1.4m,  c3.jpg) B(0.043m, 0.75m,  f6.jpg) C(0.043m, 1.2m,
c2.jpg) B(0.043m, 0.75m,  f1.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m,
0.75m,  f9.jpg) AD(0.043m, 1m,  c12.jpg)
F6 AE(0.043m, 1.4m,  c1.jpg) B(0.043m, 0.75m,  f8.jpg) C(0.043m, 1.4m,
c3.jpg) B(0.043m, 0.75m,  f3.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m,
0.75m,  f5.jpg) C(0.043m, 1.4m,  c3.jpg) B(0.043m, 0.75m,  f9.jpg)
C(0.043m, 1.2m,  c2.jpg) B(0.043m, 0.75m,  f7.jpg) C(0.043m, 1.4m,
c3.jpg) B(0.043m, 0.75m,  f8.jpg) C(0.043m, 1.4m,  c3.jpg) B(0.043m,
0.75m,  f4.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m, 0.75m,  f5.jpg)
C(0.043m, 1.4m,  c3.jpg) B(0.043m, 0.75m,  f6.jpg) C(0.043m, 1.2m,
c2.jpg) B(0.043m, 0.75m,  f3.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m,
0.75m,  f7.jpg) AD(0.043m, 1m,  c12.jpg)
F5 AE(0.043m, 1.4m,  c1.jpg) B(0.043m, 0.75m,  f5.jpg) C(0.043m, 1.4m,
c3.jpg) B(0.043m, 0.75m,  f6.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m,
0.75m,  f3.jpg) C(0.043m, 1.4m,  c3.jpg) B(0.043m, 0.75m,  f2.jpg)
C(0.043m, 1.2m,  c2.jpg) B(0.043m, 0.75m,  f5.jpg) C(0.043m, 1.4m,
c3.jpg) B(0.043m, 0.75m,  f4.jpg) C(0.043m, 1.4m,  c3.jpg) B(0.043m,
0.75m,  f8.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m, 0.75m,  f8.jpg)
C(0.043m, 1.4m,  c3.jpg) B(0.043m, 0.75m,  f7.jpg) C(0.043m, 1.2m,
c2.jpg) B(0.043m, 0.75m,  f2.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m,
0.75m,  f1.jpg) AD(0.043m, 1m,  c12.jpg)
F4 AE(0.043m, 1.4m,  c1.jpg) B(0.043m, 0.75m,  f6.jpg) C(0.043m, 1.4m,
c3.jpg) B(0.043m, 0.75m,  f3.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m,
0.75m,  f9.jpg) C(0.043m, 1.4m,  c3.jpg) B(0.043m, 0.75m,  f9.jpg)
C(0.043m, 1.2m,  c2.jpg) B(0.043m, 0.75m,  f3.jpg) C(0.043m, 1.4m,
c3.jpg) B(0.043m, 0.75m,  f8.jpg) C(0.043m, 1.4m,  c3.jpg) B(0.043m,
0.75m,  f9.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m, 0.75m,  f6.jpg)
C(0.043m, 1.4m,  c3.jpg) B(0.043m, 0.75m,  f8.jpg) C(0.043m, 1.2m,
c2.jpg) B(0.043m, 0.75m,  f9.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m,
0.75m,  f5.jpg) AD(0.043m, 1m,  c12.jpg)
F3 AE(0.043m, 1.4m,  c1.jpg) B(0.043m, 0.75m,  f7.jpg) C(0.043m, 1.4m,
c3.jpg) B(0.043m, 0.75m,  f6.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m,
0.75m,  f1.jpg) C(0.043m, 1.4m,  c3.jpg) B(0.043m, 0.75m,  f5.jpg)
C(0.043m, 1.2m,  c2.jpg) B(0.043m, 0.75m,  f6.jpg) C(0.043m, 1.4m,
c3.jpg) B(0.043m, 0.75m,  f5.jpg) C(0.043m, 1.4m,  c3.jpg) B(0.043m,
0.75m,  f4.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m, 0.75m,  f9.jpg)
C(0.043m, 1.4m,  c3.jpg) B(0.043m, 0.75m,  f8.jpg) C(0.043m, 1.2m,
```

```
c2.jpg) B(0.043m, 0.75m,  f8.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m,
0.75m,  f4.jpg) AD(0.043m, 1m,  c12.jpg)
F2 AE(0.043m, 1.4m,  c1.jpg) B(0.043m, 0.75m,  f6.jpg) C(0.043m, 1.4m,
c3.jpg) B(0.043m, 0.75m,  f8.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m,
0.75m,  f9.jpg) C(0.043m, 1.4m,  c3.jpg) B(0.043m, 0.75m,  f7.jpg)
C(0.043m, 1.2m,  c2.jpg) B(0.043m, 0.75m,  f2.jpg) C(0.043m, 1.4m,
c3.jpg) B(0.043m, 0.75m,  f6.jpg) C(0.043m, 1.4m,  c3.jpg) B(0.043m,
0.75m,  f1.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m, 0.75m,  f4.jpg)
C(0.043m, 1.4m,  c3.jpg) B(0.043m, 0.75m,  f7.jpg) C(0.043m, 1.2m,
c2.jpg) B(0.043m, 0.75m,  f9.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m,
0.75m,  f8.jpg) AD(0.043m, 1m,  c12.jpg)
F1 AE(0.043m, 1.4m,  c1.jpg) B(0.043m, 0.75m,  f5.jpg) C(0.043m, 1.4m,
c3.jpg) B(0.043m, 0.75m,  f7.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m,
0.75m,  f4.jpg) C(0.043m, 1.4m,  c3.jpg) B(0.043m, 0.75m,  f9.jpg)
C(0.043m, 1.2m,  c2.jpg) B(0.043m, 0.75m,  f9.jpg) C(0.043m, 1.4m,
c3.jpg) B(0.043m, 0.75m,  f8.jpg) C(0.043m, 1.4m,  c3.jpg) B(0.043m,
0.75m,  f4.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m, 0.75m,  f5.jpg)
C(0.043m, 1.4m,  c3.jpg) B(0.043m, 0.75m,  f3.jpg) C(0.043m, 1.2m,
c2.jpg) B(0.043m, 0.75m,  f8.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m,
0.75m,  f8.jpg) AD(0.043m, 1m,  c12.jpg)
G AE(2.5m, 1.8m,  cb1.jpg) B(2.5m, 0.6m,  f10.jpg) C(2.5m, 3.2m,
cb10.jpg) B(2.5m, 1.3m,  p11.jpg) C(2.5m, 5.2m,  cb11.jpg) B(2.5m,
0.62m,  f12.jpg) C(2.5m, 7.4m,  cb12.jpg) AE(2.5m, 1.8m,  cb1.jpg)
B(2.5m, 0.6m,  f10.jpg) C(2.5m, 11m,  cb10.jpg) B(2.5m, 1.3m,
p11.jpg) C(2.5m, 13m,  cb11.jpg) B(2.5m, 0.62m,  f12.jpg) C(2.5m, 15m,
cb12.jpg) AE(2.5m, 1.8m,  cb1.jpg) B(2.5m, 0.6m,  f10.jpg) C(2.5m,
18m,  cb10.jpg) B(2.5m, 1.3m,  p11.jpg) C(2.5m, 20m,  cb11.jpg)
B(2.5m, 0.62m,  f12.jpg) C(2.5m, 22m,  cb12.jpg) AE(2.5m, 1.8m,
cb1.jpg)
```

Annex 4: (Fitxer implícit de la figura 6 d'imatge generada)

```
T(0.043m, 24m,  t.jpg)
F AE(0.043m, 1.4m,  c1.jpg) B(0.043m, 0.75m,  f9.jpg) C(0.043m, 1.4m,
c3.jpg) B(0.043m, 0.75m,  f1.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m,
0.75m,  f9.jpg) C(0.043m, 1.4m,  c3.jpg) B(0.043m, 0.75m,  f2.jpg)
C(0.043m, 1.2m,  c2.jpg) B(0.043m, 0.75m,  f7.jpg) C(0.043m, 1.4m,
c3.jpg) B(0.043m, 0.75m,  f6.jpg) C(0.043m, 1.4m,  c3.jpg) B(0.043m,
0.75m,  f7.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m, 0.75m,  f4.jpg)
C(0.043m, 1.4m,  c3.jpg) B(0.043m, 0.75m,  f6.jpg) C(0.043m, 1.2m,
c2.jpg) B(0.043m, 0.75m,  f8.jpg) C(0.043m, 1.2m,  c2.jpg) B(0.043m,
0.75m,  f1.jpg) AD(0.043m, 1m,  c12.jpg)*10
G AE(2.5m, 1.8m,  cb1.jpg) B(2.5m, 0.6m,  f10.jpg) C(2.5m, 3.2m,
cb10.jpg) B(2.5m, 1.3m,  p11.jpg) C(2.5m, 5.2m,  cb11.jpg) B(2.5m,
0.62m,  f12.jpg) C(2.5m, 7.4m,  cb12.jpg)
```

glossari

- **Un píxel** o píxel, plural píxels (acrònim de l'anglès picture element, "element d'imatge") és la menor unitat homogènia en color que forma part d'una imatge digital, ja sigui aquesta una fotografia, un fotograma de vídeo o un gràfic.

- **Un Texel** (contracció de l'anglès texture element, o també texture píxel) és la unitat mínima d'una textura aplicada a una superfície, usada en gràfics per computador. De la mateixa manera que una imatge digital es representa mitjançant una matriu de píxels, una textura es pot representar mitjançant un matriu de texels.

- **BoundingBox**: Rectangle més petit possible que tanca completament tots els punts píxel d'un objecte.

imatges en Matlab

Matlab emmagatzema les imatges com vectors bidimensionals (matrius), al que cada element de la matriu correspon a un sol píxel.

tipus d'imatges

- Imatges indexades: **[I, MAP]**

Imatge els píxels tenen valors que són índexs directes a un mapa de color RGB. La matriu I conté la imatge indexada i la matriu **MAP** la paleta de colors RGB associada. Si la imatge és de 100x100x256 colors, la matriu **I** és de 100x100 amb rang de valors en [1...256] i la matriu **MAP** és de 256x3 amb rang de valors en [0 .. 1]. En el cas d'imatges en nivells de gris, la paleta de colors sol tenir tots els nivells de gris ordenats, de manera que en la posició **i** de la paleta esta l'i-esim nivell de gris.

- Imatges de nivells de Gris: **I**

La matriu **I** conté la imatge en nivells de gris, on el rang de valors és de [0 .. 1]. No necessita una paleta de colors. Les imatges en blanc i negre són un cas especial d'aquest tipus, on només hi ha els valors 0 i 1.

- Imatges RGB: **RGB**

Cada matriu **R, G, B**, conté les intensitats en vermell, verd i blau respectivament de la imatge

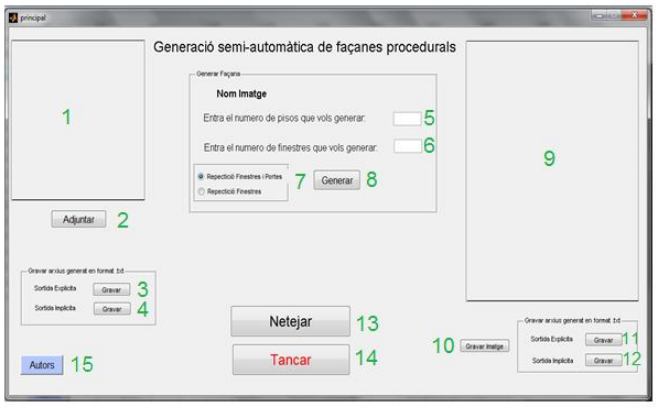
paleta: És un array de tuples de 3 enters que representa la paleta de colors (en RGB) de la imatge

Capítol 15: Manual de Usuari

En aquesta apartat s'explicarà les diferents accions que pot fer l'usuari amb l'aplicació.

15.1 Pantalla Inicial

En la pantalla inicial es on triarem les imatges per generar façanes amb diferents paràmetres. Tenim dos caixes d'imatges (1 i 9 a la figura 59), tres Panel: que incloent els elements 5 al 8 per generar fitxer explícit i implícit de la imatge d'entrada, 3 i 4 per generar fitxer explícit i implícit de la imatge generada, i l'últim 11 i 12 per omplir els paràmetres. 10 Botons (que incloent els elements 2, 3, 4, 8, 10, 11, 12, 13, 14, 15), 10 caixes de text, un botó de grup per les opcions (que inclou l'element 7) (veure figura 59).



1 - Mostrar la imatge d'entrada
2 - Botó Adjuntar imatge
3 - Botó Gravar arxiu explícit de la imatge d'entrada
4 - Botó Gravar arxiu implícit de la imatge d'entrada
5 - Escriure el número de pisos que cal generar
6 - Escriure el número de finestres per planta que cal generar
7 - Triar opció, Generar façana amb només finestres o amb finestres i portes
8 - Botó Generar façana
9 - Mostrar Imatge generada
10 - Gravar Imatge generada
11 - Botó Gravar arxiu explícit de la imatge generada
12 - Botó Gravar arxiu implícit de la imatge generada
13 - Botó Netejar imatge d'entrada i imatge generada
14 - Sortir de l'aplicació
15 - Botó per mostrar el autor i els tutors del projecte

Figura 59: Pantalla inicial de l'aplicació

15.2 Per generar un fitxer explícit de la imatge d'entrada

Per generar un fitxer .txt explícit, només ha d'adjuntar una imatge regular i prémer el botó Gravar Sortida Explicita (veure figura 60).

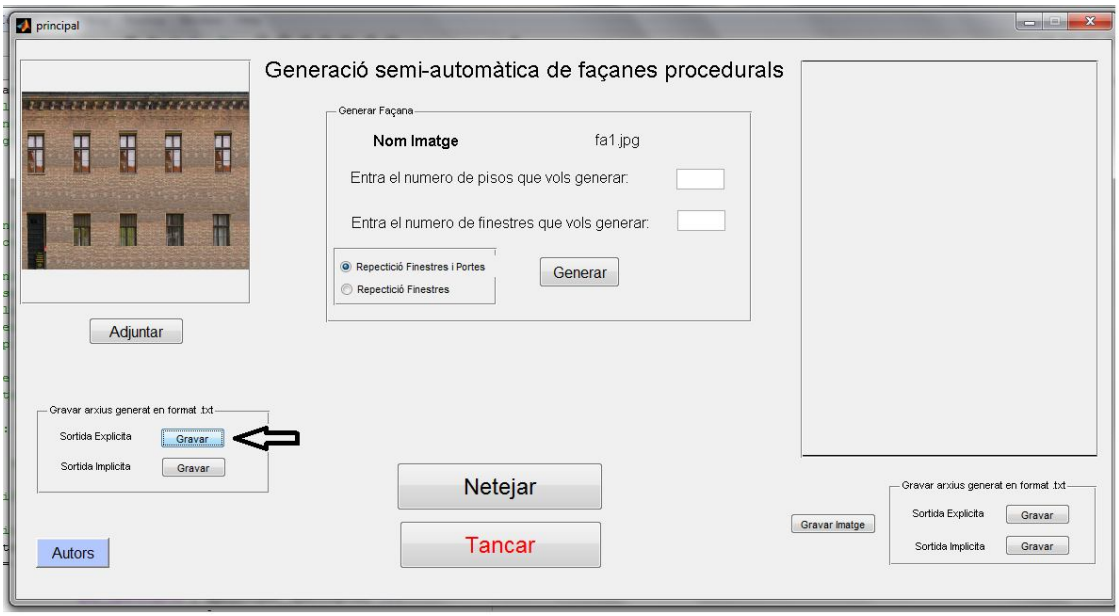


Figura 60: Finestra inicial mostra com gravar un fitxer explícit amb una façana adjuntat

15.3 Per generar un fitxer implícit de la imatge d'entrada

Per generar un fitxer .txt implícit, només ha d'adjuntar una imatge regular i prémer el botó Gravar Sortida Implícita (veure figura 61).

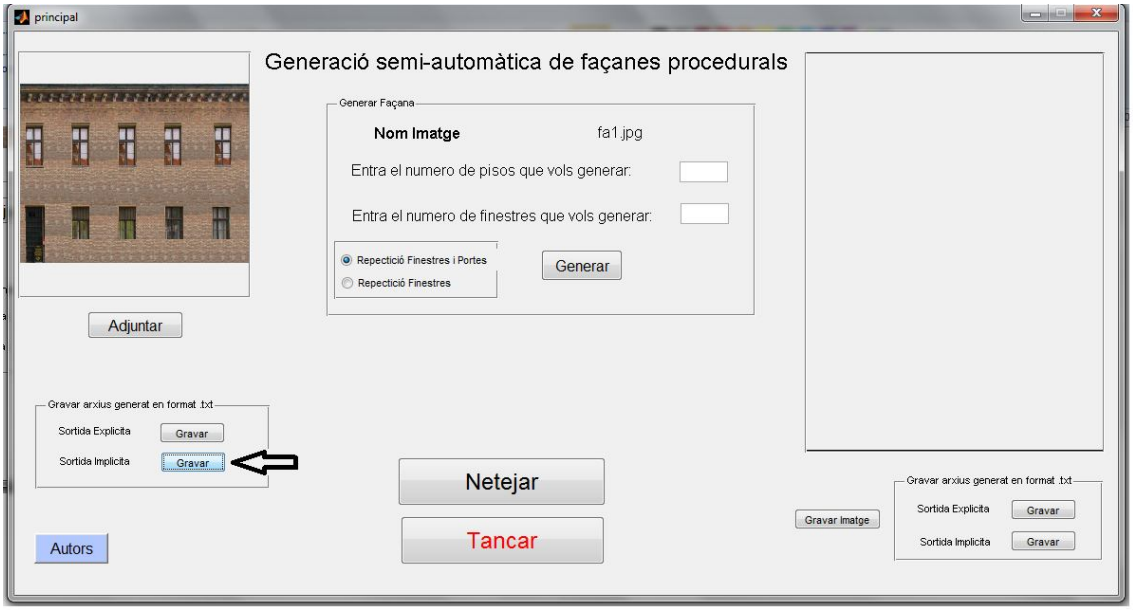


Figura 61: Finestra inicial mostra com gravar un fitxer implícit amb una façana adjuntat

15.4 Per generar façana amb diferents paràmetres d'entrada

Per generar una façana s'ha d'entrar el número de pisos i el número de finestres per planta que cal generar. A continuació s'ha de triar una opció: generar façana amb repeticions de finestres o repeticions de finestres i portes. Després prémer el botó Generar (veure figures 62 i 63).

Figura 62: Imatge generat amb opció Repeticions Finestres i portes

Figura 63: Imatge generat amb opció Repeticions Finestres

15.5 Per guardar la imatge generada

Un cop premut el botó Generar, l'aplicació mostra la imatge generada. Per tant per guardar aquesta imatge només ha de prémer el botó "Gravar Imatge" (veure figura 64).

Figura 64: El pas per gravar la imatge generada

15.6 Per generar un fitxer explícit de la imatge generada

Després de generar la façana, és pot generar el fitxer .txt explícit de la imatge generada, prement el botó Gravar Sortida Explicita (veure figura 65).

Figura 65: Gravar fitxer explícit de la imatge generada

15.7 Per generar un fitxer implícit de la imatge generada

Finalment per generar un fitxer implícit de la imatge generada, s'ha de prémer el botó Gravar Sortida Implícita (veure figura 66).

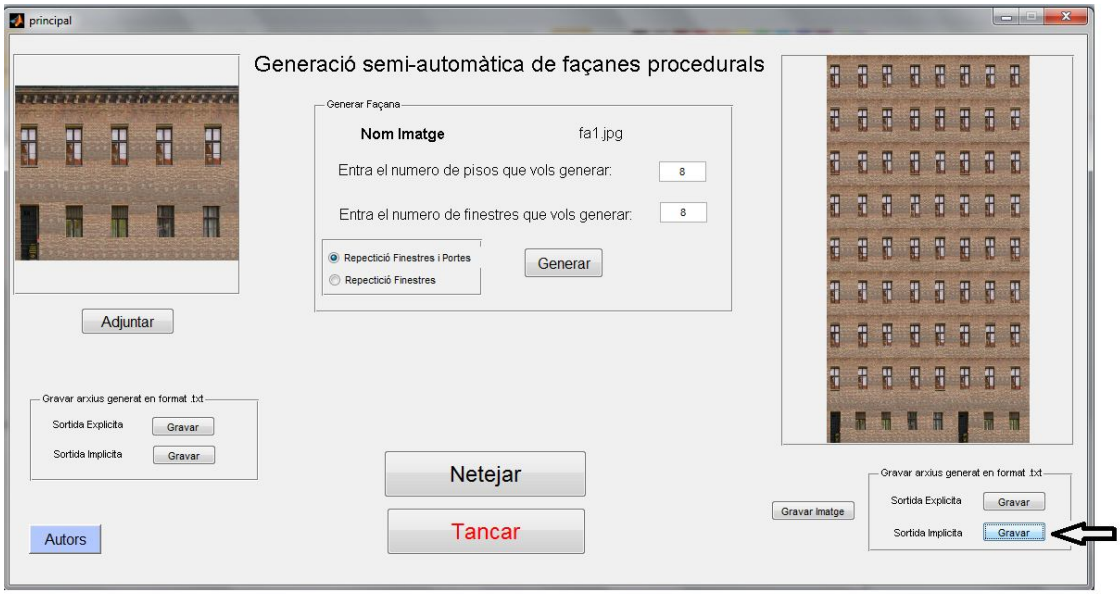


Figura 66: Gravar fitxer implícit de la imatge generada

15.8 Per veure el autor i els tutors del projecte

Aquest opció es pot saber en qualsevol moment, només prem el botó Autors, es mostra la figura 67.

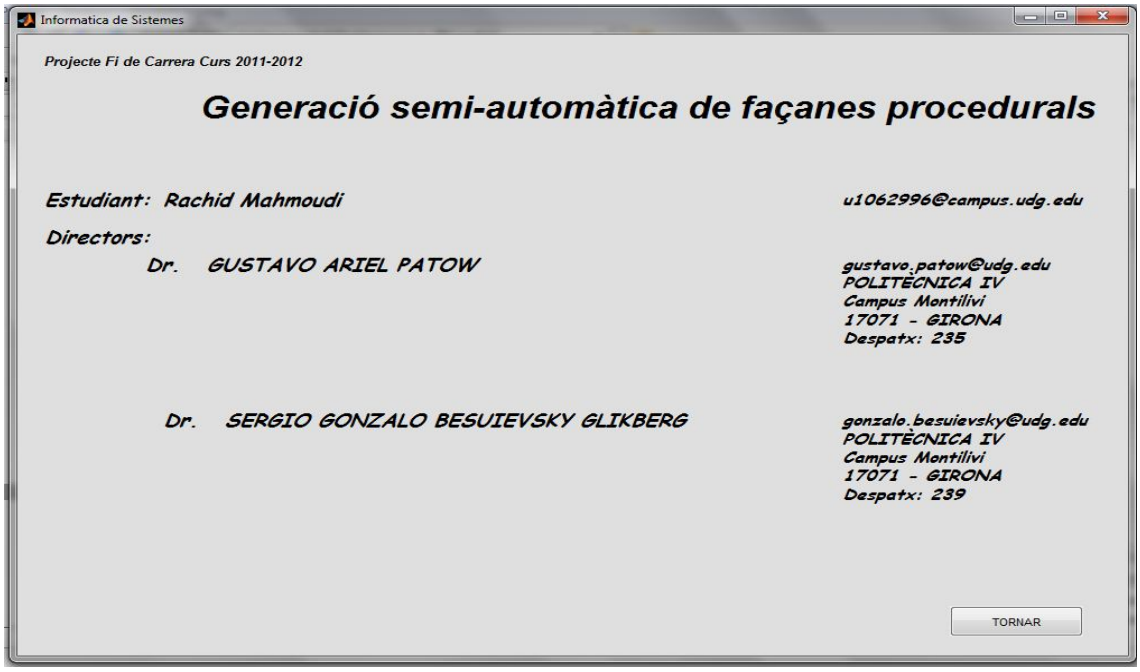


Figura 67: pagina on es mostra el títol, l'autor, i els tutors del projecte